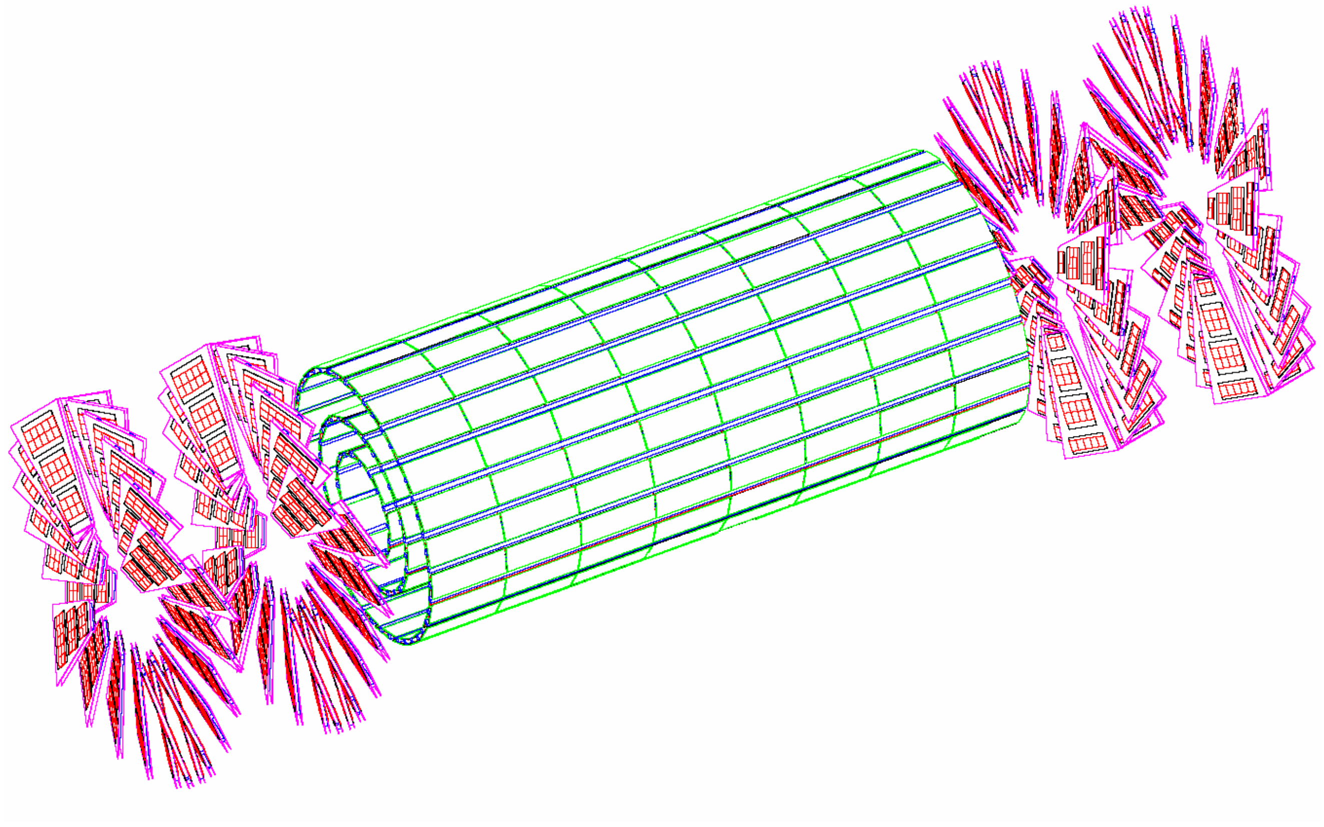
CMS Pixel Database

Detector Framework



Brief Introduction

- The CMS Pixel Detector consists of a cylindrical central detector with 3 barrel layers
    - Total 3 (initially 2)
    - z = ± 266.6 mm
    - r = 41.05 mm – 104.26 mm

- and 2 forward detectors, one at either end of the barrel, each containing 3 disks

    - Total 6 (initially 4)
    - $z = \pm$ 34.5 cm, $\pm$ 46.5 cm, and 57.4 cm
    - r_{min} = 6 cm
    - r_{max} = 15 cm

- Total coverage in rapidity of the pixel detector is $-2.5 < \eta < 2.5$

- Each readout chip reads out 4160 pixels, a pixel being 100 μm x 150 μm in size. This is the CMOS 0.25 μm version of the chip with 52 columns and 80 rows.

Central Pixel Detector

- The central pixel detector consists of 3 barrels.
- A barrel is made up of rectangular modules connected together lengthwise to form a ladder.
- A ladder consists of 8 such modules.
- A module has a sensor bonded to 8 readout chips (ROCs) mounted on each face for a total of 16 ROCs.

Barrel 1

- In layer 1 there are 16 full ladders and 4 half-ladders, equivalent to 18 full ladders.
- The total number of modules is 128 full modules and 32 half modules, equivalent to 144 full modules.
- Total of 18 full ladders equivalent
- Total number of modules: 144 (full modules equivalent)

Total number of ROC's in barrel 1: 2304
Total number of sensors: 160

Barrel 2

- 28 full ladders + 4 half ladders ≡ 30 full ladders equivalent
- 224 full modules & 32 half modules
- Total number of modules: 30 x 8 = 240 (full modules equivalent)

Total number of ROC's: 240 x 16 = 3840
Total number of sensors: 256

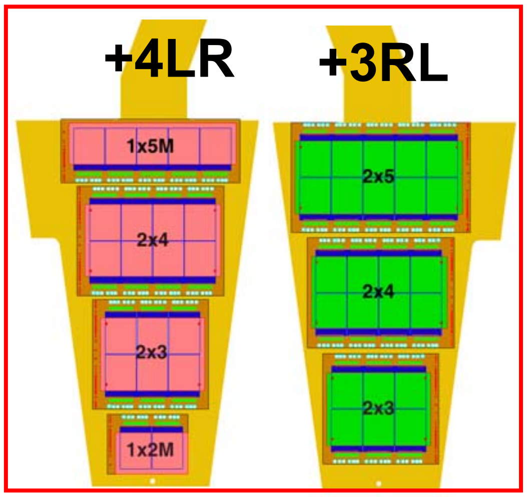
Barrel 3

- 40 full ladders + 4 half ladders ≡ 42 full ladders equivalent
- 320 full modules & 32 half modules
- Total number of modules: 42 x 8 = 336 (full modules equivalent)

Total number of ROC's: 336 x 16 = 5376
Total number of sensors: 352

Forward Pixel Detector:Disks

There are 2 sets of forward disks, one at either end of the central detector. Each disk is made up of 24 "turbine" blades. The blades are rotated 20^0 counterclockwise about its vertical axis. Each blade consists of 2 panels, one on each face. The panels on the face of the disk facing the IR contain 4 plaquettes while those on the other side contain 3 plaquettes. The different plaquettes contain sensors bonded to multiple readout chips as shown in the figure below. The sensors are mounted in such a manner that the active



regions on the two sides overlap.

Number of plaquettes (4 disks): (4 x 24 x 4 + 3 x 24 x 4) = 672
Number of sensors: 672
Number of plaquettes (6 disks): (4 x 24 x 6 + 3 x 24 x 6) = 1008
Number of sensors: 1008

A blade consists of a total of 45 ROCs.

Number of ROC's (4 disks): 45 x 96 = 4320
Number of ROC's (6 disks) = 4320 + 1660 = 5980

Total Number of ROC's (2 barrels + 4 disks): 10,464
Total Number of ROC's (3 barrels + 4 disks): 15,840
Total Number of ROC's (3 barrels + 6 disks): 17,500

- Number of pixels in a ROC (0.25 μm) = 52 x 80 = 4160

Total number of pixels

10464 x 4160 = 43530240	**~ 44M**	**(2 barrels + 4 disks)**
15840 x 4160 = 65894400	**~ 66M**	**(3 barrels + 4 disks)**
17500 x 4160 = 72800000	**~ 73M**	**(3 barrels + 6 disks)**

Pixel size: 100 x 150 μm (0.25μm CMOS)

Detector Numbering Scheme (Logical)

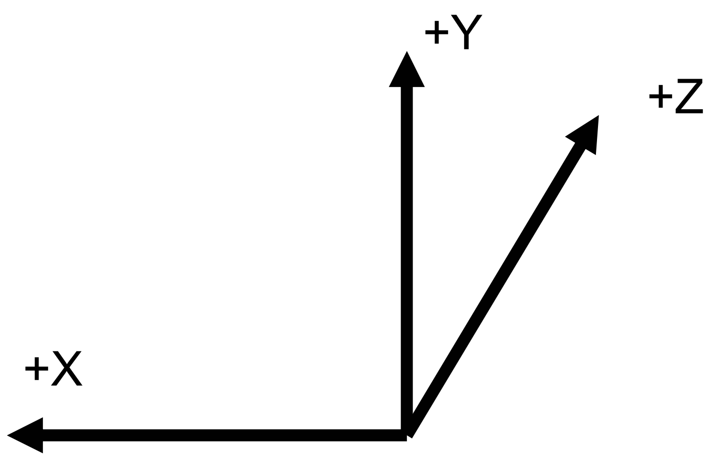
For convenience, separate all geometry information into two groups

- Physical – to present the detector as constructed
- Logical – to present the detector in a more convenient form for storing detector related information

Adopt a coordinate system in which the +z is along the direction of the B field of the solenoid, i.e. +B. For reference purposes, assume the +B direction points into the plane of display as shown in the figure. It follows then that +x is left and +y is up. This is the standard CMS coordinate system.

The official CMS coordinate system is :

+X:	horizontal, in the direction of the LHC center,
+Y:	up
+Z:	towards Jura Mountains (B field is in the +z direction)



Barrel Numbering Scheme

Barrels

- There are 3 barrels in all

Inner barrel:	ID 0
Middle barrel:	ID 1
Outer barrel:	ID 2

Shells

- Mechanically, the barrel consists of 4 shells (quarter)
- Divided by the x-y plane at z=0 into +z and –z halves

Barrel Long	+z side	ID: 0
Barrel Long	–z side	ID: 1

- Divided by the y-z plane at x=0 into left/right halves (left/right half cylinder)

Barrel Trans	+x side	ID: 0
Barrel Trans	–x side	ID: 1

- (Long, Trans) of shells:

Shell (0, 0)	ID: 0	(+z, +x)
Shell (0, 1)	ID: 1	(+z, -x)
Shell (1, 0)	ID: 2	(-z, +x)
Shell (1, 1)	ID: 3	(-z, -x)

Sectors

- Each shell is organized into 8 sectors in phi
- A total of 32 sectors – 16 on the +z side (P) and 16 on the –z side (N).
- On the P side (+z), sector 0 is on the left side (+x side (+) with observer standing at collision point) and at the top (of vertical slice made by t he y-z plane) while 15 is on the top right side (-x side (-)).

  Sector ID: 0 (+1P)
  Sector ID: 1 (+2P)
  ⋮
  Sector ID: 7 (+8P)
  Sector ID: 8 (-8P)
  ⋮
  Sector ID:15 (-1P)

- On the N side (-z), sector 0 is on the right side (+x side with observer standing at the collision point) and at the top while 15 is on the top left side (-x side).

  Sector ID: 0 (+1N)
  Sector ID: 1 (+2N)
  ⋮
  Sector ID: 7 (+8N)
  Sector ID: 8 (-8N)
  ⋮
  Sector ID: 15 (-1N)

Legend:

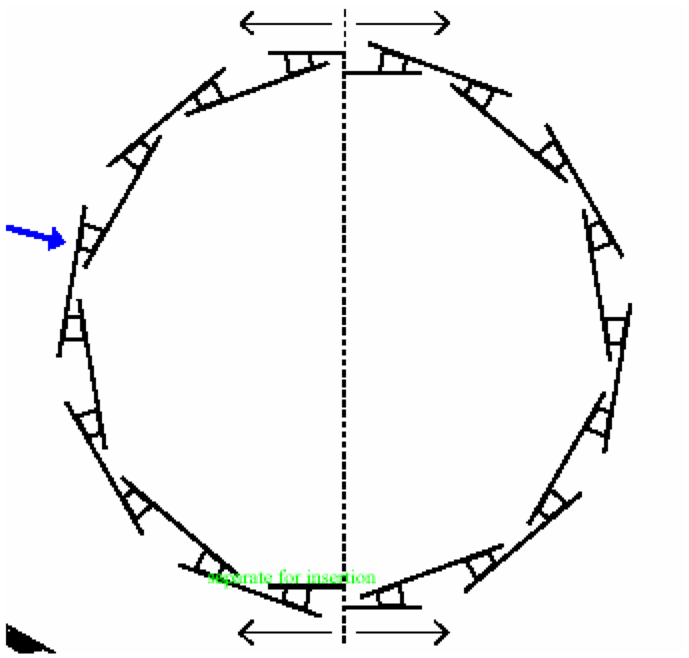
P ➔ +Z side
N ➔ -Z side
+ ➔ +X side
- ➔ -X side

Sector numbering:

- o Increasing counterclockwise starting at X=0 and Y > 0 for an observer facing in the +Z direction.

Ladders

- A barrel is made up of ladders arranged as shown in the figure below.
- The plates connected to the two neighbouring ladders are the cooling plates.
- The vertical dotted line represents the YZ plane that enables the barrel to be separated for insertion.



- The inner barrel consists of 16 full (F) and 4 half (H) ladders. The ID number starts at top (left half (+X direction) as observed when standing at collision point and facing in the +Z direction) and increase counter-clockwise.

Ladder	ID: 0	Type: H	(+1H)	➔top ½ ladder in left half (+X side)
Ladder	ID: 1	Type: F	(+2F)	➔2nd highest in left half
Ladder	ID: 2	Type: F	(+3F)	➔3rd
⋮				
Ladder	ID: 8	Type: F	(+9F)	
Ladder	ID: 9	Type: H	(+10H)	➔bottom in left half (+X)
Ladder	ID: 10	Type: H	(-10H)	➔bottom in right half (-X)
Ladder	ID: 11	Type: F	(-9F)	
⋮				
Ladder	ID: 18	Type: F	(-2F)	➔2nd highest in right half (-X side)
Ladder	ID: 19	Type: H	(-1H)	➔topmost in right half (-X side)

- The middle barrel consists of 28 F and 4 H ladders. The ID number starts at top (left half) and increase counter-clockwise.

Ladder	ID: 0	Type: H	(+1H)	➔top ½ ladder in +X side
Ladder	ID: 1	Type: F	(+2F)	➔2nd highest in +X side
Ladder	ID: 2	Type: F	(+3F)	➔3rd
⋮				
Ladder	ID: 14	Type: F	(+15F)	
Ladder	ID: 15	Type: H	(+16H)	➔bottom in +X side
Ladder	ID: 16	Type: H	(-16H)	➔bottom in –X side
Ladder	ID: 17	Type: F	(-15F)	
⋮				
Ladder	ID: 30	Type: F	(-2F)	➔2nd highest in -X side
Ladder	ID: 31	Type: H	(-1H)	➔topmost in -X side

- The outer barrel consists of 40 F and 4 H ladders

Ladder	ID: 0	Type: H	(+1H)	➔top ½ ladder in +X side
Ladder	ID: 1	Type: F	(+2F)	➔2nd highest in –X side
Ladder	ID: 2	Type: F	(+3F)	➔3rd
⋮				
Ladder	ID: 20	Type: F	(+21 F)	
Ladder	ID: 21	Type: H	(+22 H)	➔bottom in +X side
Ladder	ID: 22	Type: H	(-22 H)	➔bottom in –X side
Ladder	ID: 23	Type: F	(-21 F)	
⋮				
Ladder	ID: 42	Type: F	(-2F)	➔2nd highest in -X side
Ladder	ID: 43	Type: H	(-1H)	➔topmost in -X side

	Barrel 1	Barrel 2	Barrel 3
Radius (mm)	41.05 - 46.46	70.16-75.55	98.88-104.26
Faces full/half (full equiv)	16/2	28/2	40/2
Det. modules full/half	128/16 = 144	224/16 = 240	320/16 = 336
Readout chips	2304	3840	5376
Pixels (100μm x 150μm)	9.6M	16M	22.4M
Readout links	288	480	352

- A barrel ladder consists of 8 detector modules (16.2 mm width x 66.3mm length) with 0.4mm gaps in between.
  Total barrel length = (66.3 x 8 + 0.4 x 7) mm = 2x266.6 mm = 533.2 mm
- The effective gaps are larger since there is a 0.9mm inactive area on each sensor (used for isolation rings).
- The total gap between 2 sensors is (in z) 0.4+2*0.9=2.2mm.

Modules

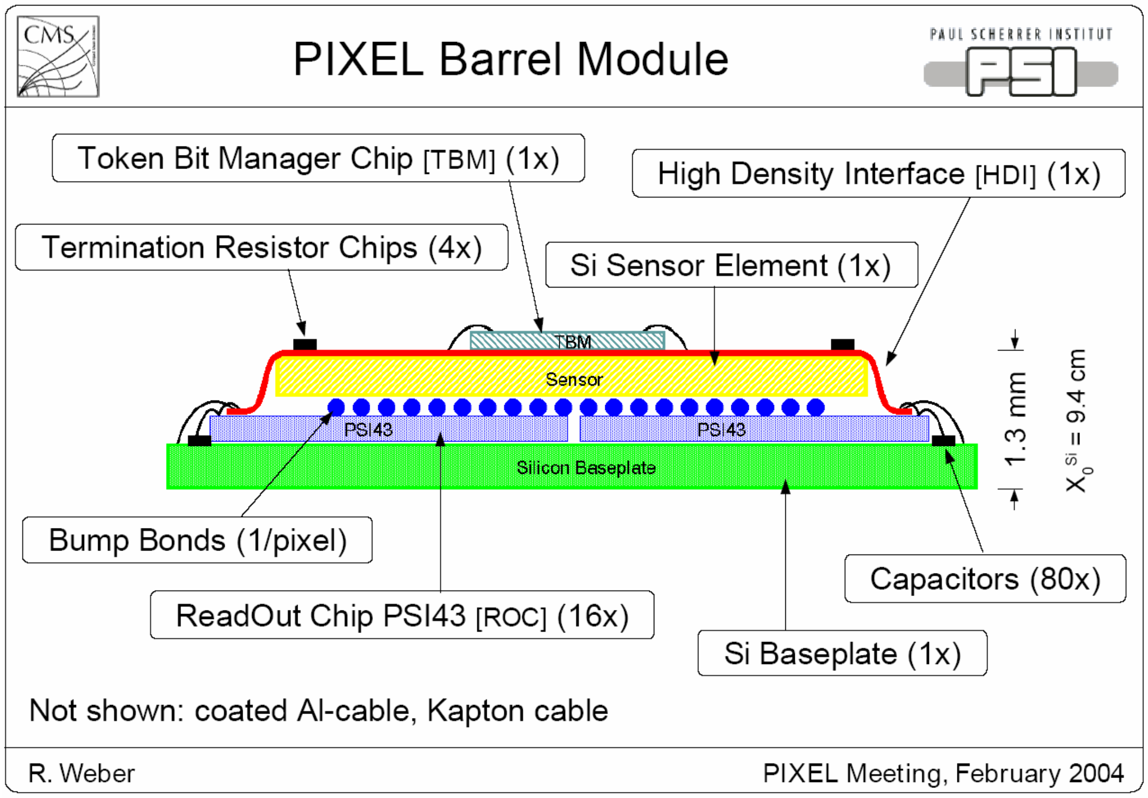
- There are 8 modules (half modules) in a ladder (half ladder)

Module	ID: -4	(farthest from IR in –Z direction)
Module	ID: -3	
⋮		
Module	ID: 3	
Module	ID: 4	(farthest from IR in +Z direction)

- A module could either be a full module (F) which contains 16 ROCs or a half module (H) which contains 8 ROCs.

Module Type: Full (F)	ID: 0
Module Type: Half (H)	ID: 1

- Each (F) module consists of a Si base plate on which resides 16 ROCs bump-bonded to a 300 μm thick Si sensor (as shown below). A half module is constructed the same way except it has 8 ROCs.


PIXEL Barrel Module
PAUL SCHERRER INSTITUT
PSI
CMS
Token Bit Manager Chip [TBM] (1x)
High Density Interface [HDI] (1x)
Termination Resistor Chips (4x)
Si Sensor Element (1x)
TBM
Sensor
PSI43
PSI43
Silicon Baseplate
1.3 mm
X0Si = 9.4 cm
Bump Bonds (1/pixel)
Capacitors (80x)
ReadOut Chip PSI43 [ROC] (16x)
Si Baseplate (1x)
Not shown: coated Al-cable, Kapton cable
R. Weber
PIXEL Meeting, February 2004

- A full (F) module has 16 ROCs and a half (H) module has 8 ROCs.
- There are 2 TBMs per module in layers 1 and 2, and 1 TBM per module in layer 3
- The TBM unit is a dual chip module. In layers 1 and 2, both chips in the module are used; whereas in layer 3 only one of the 2 chips is used.
- There is
    - 1 HDI per module
    - 1 sensor per module.

- In modules that have 2 TBMs, TBM 0 reads out ROCs 0 through 7 and TBM 1 reads out ROCs 8 through 15.

Discussion: Shells, Sectors, Ladders, etc.

- The barrel and the disks are mechanically divided into two halves (half cylinders) by the YZ plane (at X=0) into the left half and the right half. After installing each detector half inside a service half cylinder, the two halves are brought together for the final configuration.
- In the case of the barrel, the detector is logically further divided by the XY plane (at Z=0) for other purposes like HV cabling, etc. into a total of four "shells" – 2 shells in the +z half and 2 in the –z half.
- A barrel is physically constructed of ladders that run the entire length of the detector, i.e. ± 266.6 mm so that a physical ladder is "shared" by two logical shells and sectors. For example ladder +3F in barrel 0 is shared by shells 0 and 2. Modules -4 to -1 of this ladder reside in shell 2 while modules 1 to 4 reside in shell 0.
- A shell is further divided into 8 sectors so that there are in all 32 sectors.
- Just as in the case of a shell, a ladder is also shared by two sectors. In the case of ladder +3F in barrel 0, it is shared by sector 1 in the +Z and sector 1 in the –Z halves as tabulated in the table in the next page.
- A sector (as shown in the table) consists of several ladders. For example, sector 5 consists of:
    - ladder 6 in barrel 0
    - ladders 10 and 11 in barrel 1
    - ladders 13, 14, 15 in barrel 2

- Explanation of the symbols used in the table below

+1P

+ indicates that the sector is in the +X half of the barrel
- indicates that the sector is in the –X half of the barrel
1 is the sector number
P indicates that the sector is in the +Z half of the barrel
N indicates that the sector is in the –Z half of the barrel

+1H, **+2F**

+ (-) indicates that the ladder is in the +X (-X) half of the barrel
1 (2) is the ladder number
H (F) indicates that the ladder is a half (full) ladder.

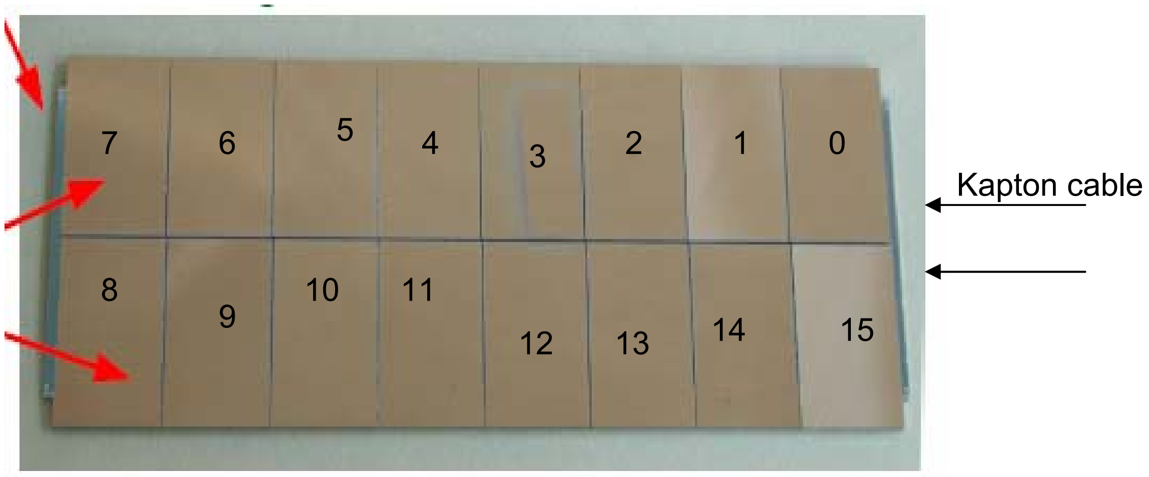
(0, 1) represent the logical numbers of the ladders in the sector of the barrel. In this example ladders 0 and 1 reside in sector 0 of barrel 0.

Sector Number	Z Location	Sector ID	Barrel 0 Ladders	Barrel 1 Ladders	Barrel 2 Ladders
0	+z	+1P	+1H,+2F (0,1)	+1H,+2F (0,1)	+1H,+2F,+3F (0,1,2)
1	+z	+2P	+3F (2)	+3F,+4F (2,3)	+4F,+5F,+6F (3,4,5)
2	+z	+3P	+4F (3)	+5F,+6F (4,5)	+7F,+8F,+9F (6,7,8)
3	+z	+4P	+5F (4)	+7F,+8F (6,7)	+10F,+11F (9,10)
4	+z	+5P	+6F (5)	+9F,+10F (8,9)	+12F,+13F (11,12)
5	+z	+6P	+7F (6)	+11F,+12F (10,11)	+14F,+15F,+16F (13,14,15)
6	+z	+7P	+8F (7)	+13F,+14F (12,13)	+17F,+18F,+19F (16,17,18)
7	+z	+8P	+9F,+10H (8,9)	+15F,+16H (14,15)	+20F,+21F,+22H (19,20,21)
8	+z	-8P	-9F,-10H (10,11)	-15F,-16H (16,17)	-20F,-21F,-22H (22,23,24)
9	+z	-7P	-8F (12)	-13F,-14F (18,19)	-17F,-18F,-19F (25,26,27)
10	+z	-6P	-7F (13)	-11F,-12F (20,21)	-14F,-15F,-16F (28,29,30)
11	+z	-5P	-6F (14)	-9F,-10F (22,23)	-12F,-13F (31,32)
12	+z	-4P	-5F (15)	-7F,-8F (24,25)	-10F,-11F (33,34)
13	+z	-3P	-4F (16)	-5F,-6F (26,27)	-7F,-8F,-9F (35,36,37)
14	+z	-2P	-3F (17)	-3F,-4F (28,29)	-4F,-5F,-6F (38,39,40)
15	+z	-1P	-1H,-2F (18,19)	-1H,-2F (30,31)	-1H,-2F,-3F (41,42,43)
0	-z	+1N	+1H,+2F (0,1)	+1H,+2F (0,1)	+1H,+2F,+3F (0,1,2)
1	-z	+2N	+3F (2)	+3F,+4F (2,3)	+4F,+5F,+6F (3,4,5)
2	-z	+3N	+4F (3)	+5F,+6F (4,5)	+7F,+8F,+9F (6,7,8)
3	-z	+4N	+5F (4)	+7F,+8F (6,7)	+10F,+11F (9,10)

4	-z	+5N	+6F (5)	+9F,+10F (8,9)	+12F,+13F (11,12)
5	-z	+6N	+7F (6)	+11F,+12F (10,11)	+14F,+15F,+16F (13,14,15)
6	-z	+7N	+8F (7)	+13F,+14F (12,13)	+17F,+18F,+19F (16,17,18)
7	-z	+8N	+9F,+10H (8,9)	+15F,+16H (14,15)	+20F,+21F,+22H (19,20,21)
8	-z	-8N	-9F,-10H (10,11)	-15F,-16H (16,17)	-20F,-21F,-22H (22,23,24)
9	-z	-7N	-8F (12)	-13F,-14F (18,19)	-17F,-18F,-19F (25,26,27)
10	-z	-6N	-7F (13)	-11F,-12F (20,21)	-14F,-15F,-16F (28,29,30)
11	-z	-5N	-6F (14)	-9F,-10F (22,23)	-12F,-13F (31,32)
12	-z	-4N	-5F (15)	-7F,-8F (24,25)	-10F,-11F (33,34)
13	-z	-3N	-4F (16)	-5F,-6F (26,27)	-7F,-8F,-9F (35,36,37)
14	-z	-2N	-3F (17)	-3F,-4F (28,29)	-4F,-5F,-6F (38,39,40)
15	-z	-1N	-1H,-2F (18,19)	-1H,-2F (30,31)	-1H,-2F,-3F (41,42,43)

ROC Indexing for Barrel Modules

- The ROC indices (0-15) follow the token scan for single TBM modules as shown in the following figure. The module is viewed from the top (TBM side).
- Modules with 2 TBMs (inner and middle barrels) have 2 readout links, one per TBM, while modules in the outer barrel (with 1 TBM) have one readout link.



- The ROCs are numbered in the order in which the token is passed.

ROC Number 0	ID: 0
ROC Number 1	ID: 1
⋮	
ROC Number 14	ID: 14
ROC Number 15	ID: 15

Forward Pixel Detector Numbering Scheme

Forward Detectors

- There are 2 forward detectors

Detector in the +Z direction:	ID 0
Detector in the –Z direction:	ID 1

Half Cylinders

- There are 2 half cylinders (relative to the +B direction)

Right half cylinder (-x):	ID 0
Left half cylinder (+x):	ID 1

Disks

- Each detector has 3 physical disks

Disk closest to IR:	ID 0
Disk (middle):	ID 1
Disk (farthest):	ID 2

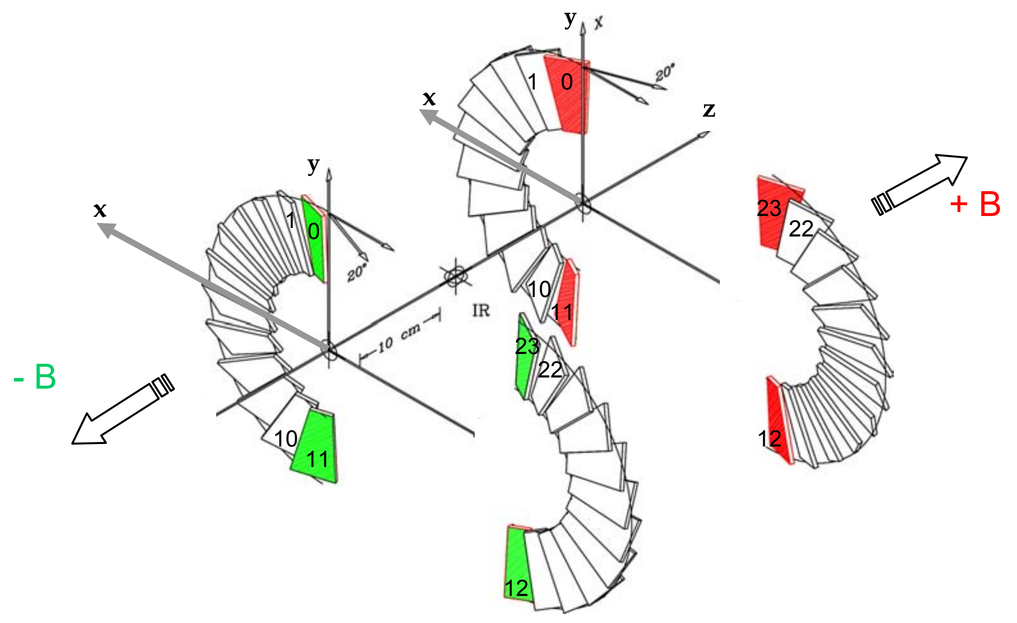
- Each disk has detector elements (sensors, readout chips, etc.) mounted on both faces.
- It is possible to identify each face of a disk, e.g. 0 for face closer to IR and 1 for the other face.
- A convenient choice is to define a face of a physical disk as 1 logical disk.
- In such a scheme, a forward pixel detector will have 6 logical disks.

Physical disk 0	
LDisk (Disk 0) closest to IR:	ID 0
LDisk (Disk 0):	ID 1
Physical disk 1	
LDisk (Disk 1, face closer to IR):	ID 2
LDisk (Disk 1):	ID 3
Physical disk 2	
LDisk (Disk 2, face closer to IR):	ID 4
LDisk (Disk 2):	ID 5

Blades (within each disk)

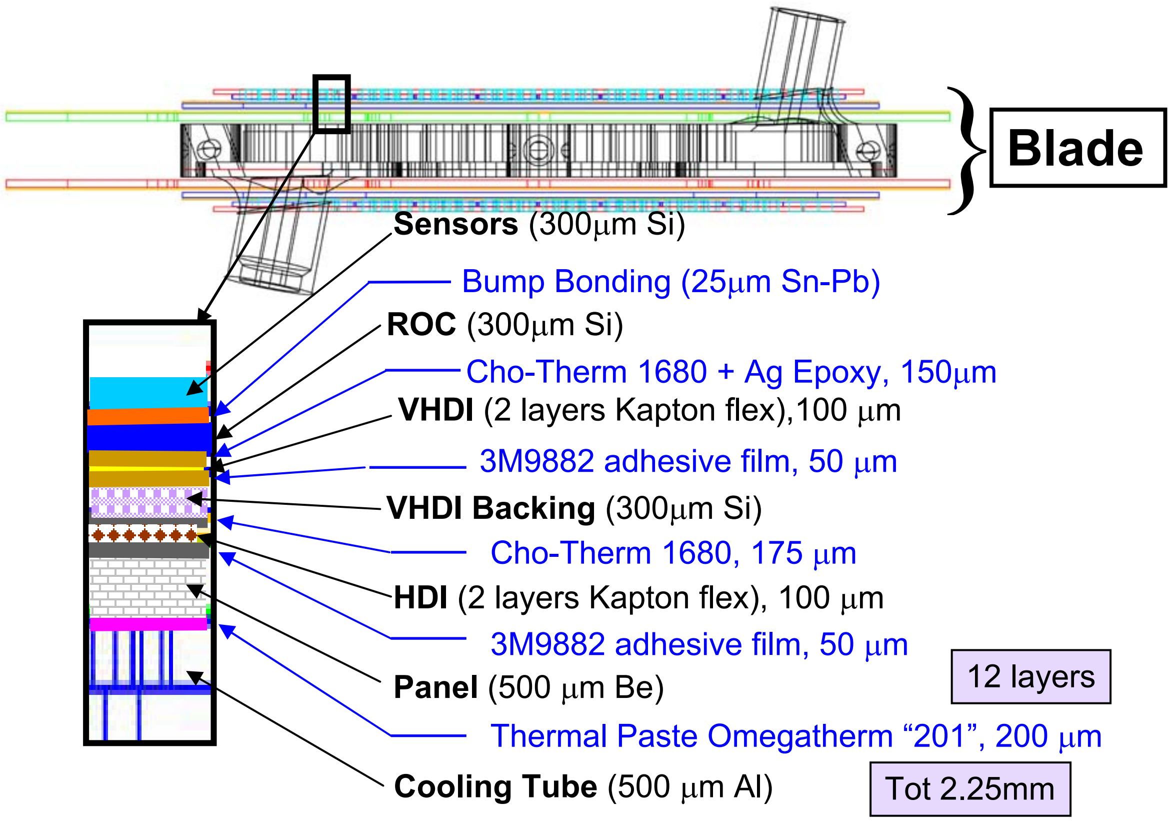
- Each physical disk has 24 turbine blades.
- The blades within a disk are numbered from 0 to 23 as shown in the figure.
- A scheme has been chosen where the numbers increase in a clockwise manner for an observer at the IR facing the disks.

Assembly



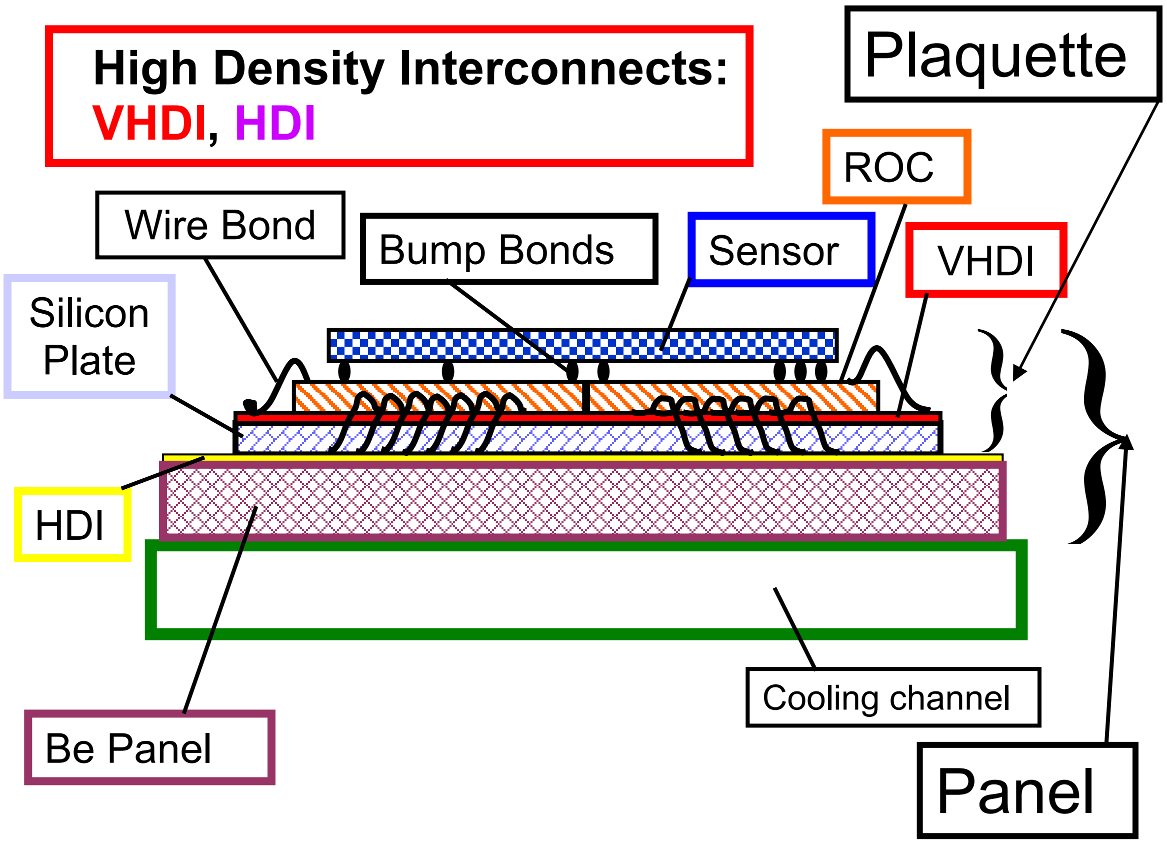
Panel/Blade Numbering Scheme

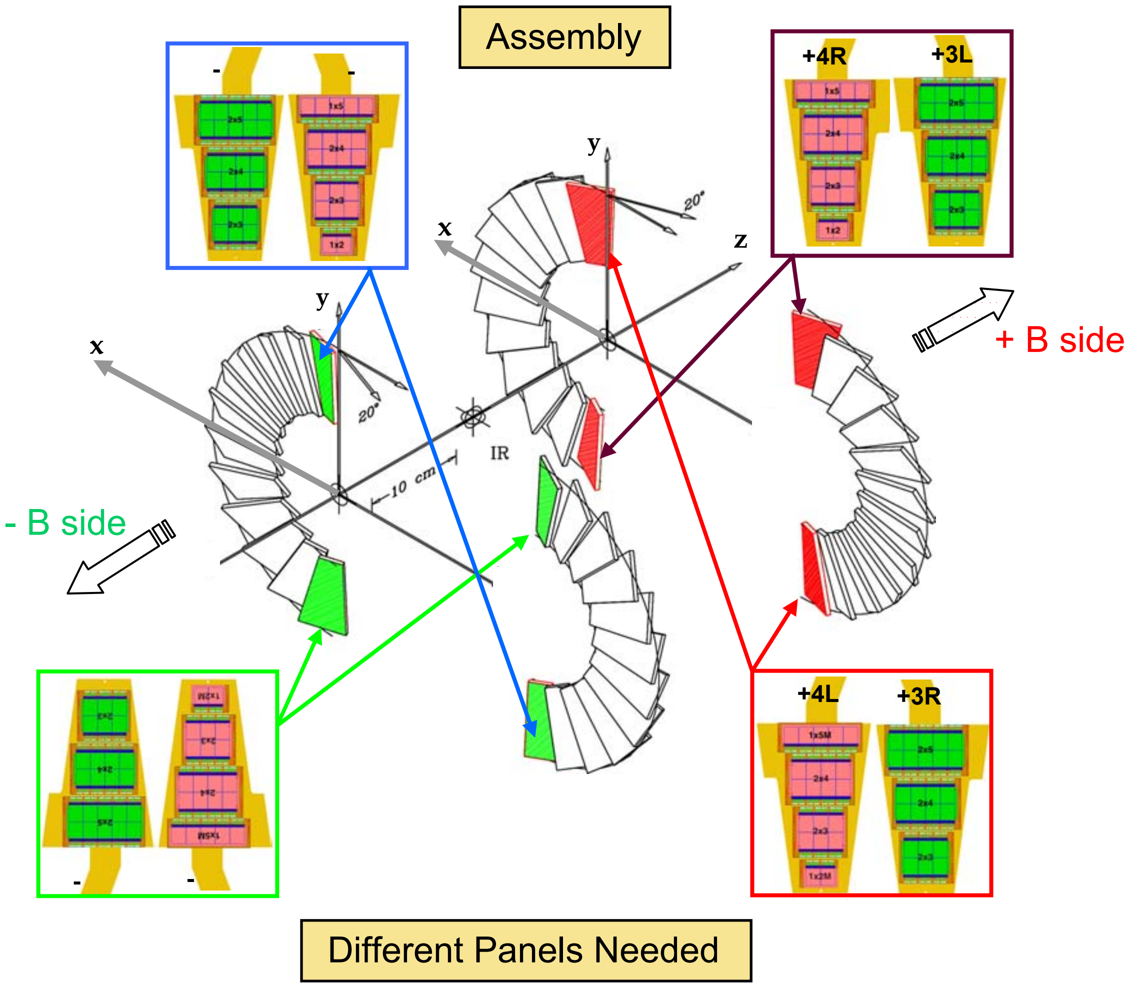
- The following diagram illustrates a cross-sectional view of a blade.


Blade
Sensors (300µm Si)
Bump Bonding (25µm Sn-Pb)
ROC (300µm Si)
Cho-Therm 1680 + Ag Epoxy, 150µm
VHDI (2 layers Kapton flex),100 µm
3M9882 adhesive film, 50 µm
VHDI Backing (300µm Si)
Cho-Therm 1680, 175 µm
HDI (2 layers Kapton flex), 100 µm
3M9882 adhesive film, 50 µm
Panel (500 µm Be)
Thermal Paste Omegatherm "201", 200 µm
Cooling Tube (500 µm Al)
12 layers
Tot 2.25mm

Panel (on a blade)

- Each blade has 2 panels, one on each side.
- Each panel has a HDI
- Could have a scheme in which a panel has the same number as the blade it resides on and an index to indicate whether the panel is closer to or farther from the IR, e.g. 0 to indicate the panel is closer to the IR and 1 to indicate it is on the other side.
- With a logical disk numbering scheme, only the panel (blade) number within the logical disk is needed.
- The following diagram illustrates a cross-sectional view of a panel.





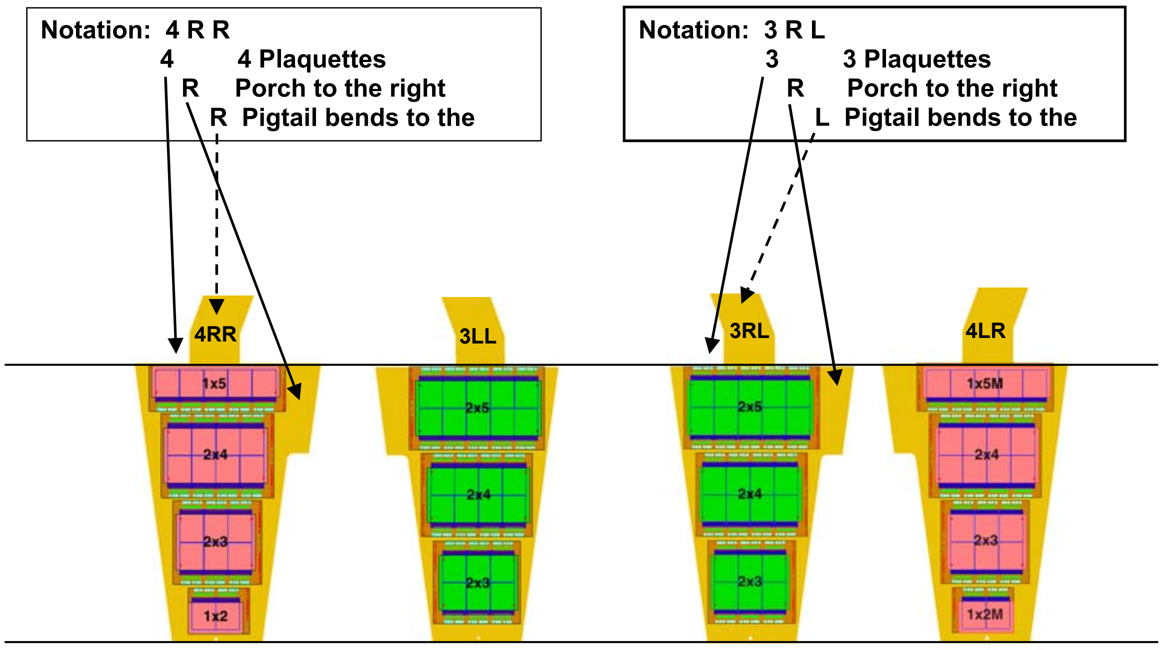
- The reference system indicated here is incorrect. The correct one should have the XY plane rotated 90^0 counterclockwise (about the Z axis).

- There are 8 different panel types (or 4 different pairs).
- 2 pairs, (+4LR, +3RL) and (+4RR, +3LL), reside on the +Z disks.
- 2 pairs, (-3LR, -4RL) and (-3RR, -4LL), reside on the –Z disks.

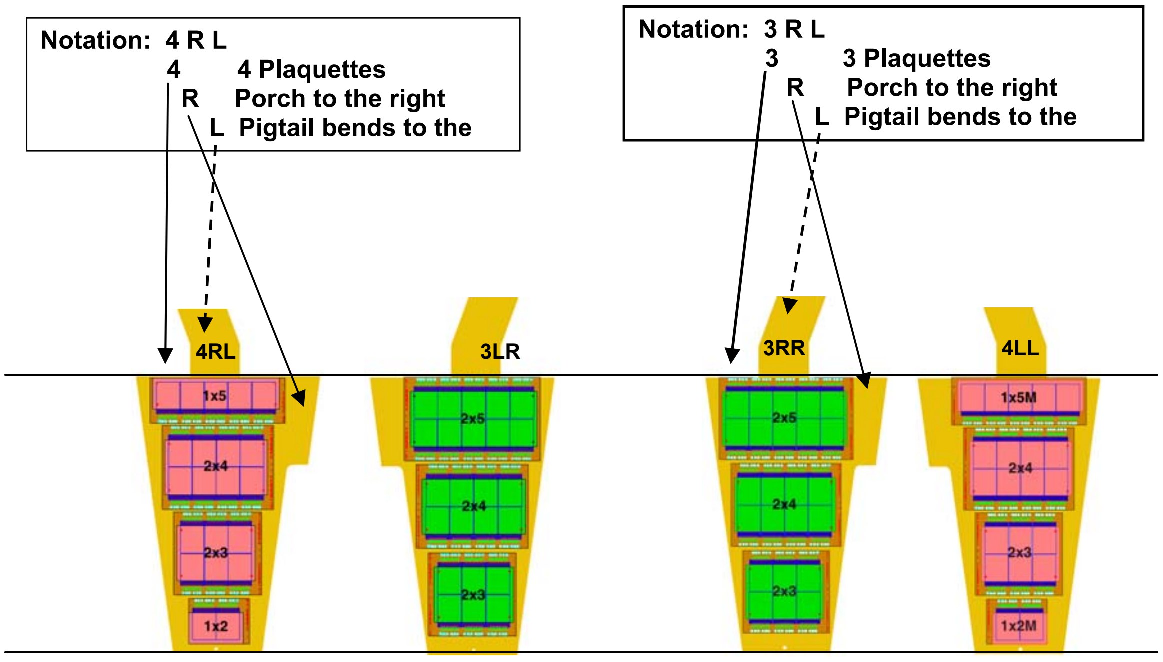
- Legend: +4LR
    - + ➔ +Z side of the IR
    - 4 ➔ panel has 4 plaquettes
    - L ➔ (first letter) porch (stub on the panel) is on the left side of panel (reference: panel viewed with pig tail up)
    - R ➔ (second letter) pig tail bends to the right (reference: panel viewed with pig tail up)

- Observations
  - In the +Z side all panels facing the IR have the pig tails bending to the right (hence pig tails of panels on the opposite side bend to the left)
  - In the –Z side all panels facing the IR have the pig tails bending to the left (hence pig tails of panels on the opposite side bend to the right)..

Notations Panels and HDIs
For Disks on the **+ Side**



Notations Panels and HDIs
For Disks on the - Side



Possible Numbering Schemes

- Two numbering schemes are possible.
- One that uses physical disk number as follows

  Forward Detector ID (0 or 1)
    Disk ID (0, 1, or 2)
      Blade ID (0, 1, . . ., 23)
        Panel ID (0 or 1)

- The other that uses logical disk numbers as follows

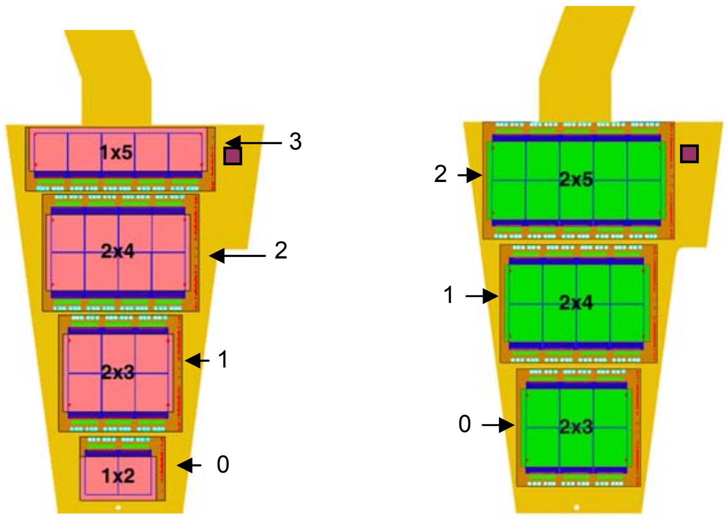
  Forward Detector ID (0 or 1)
    Logical Disk ID (0 to 5)
      Panel ID (0 to 23)

- Mapping from logical to physical disks will be straightforward

Plaquettes (Within a Panel)

- A panel contains several plaquettes.
- Each plaquette has a VHDI
- Some contain 4, number 0→3 (as shown in figure, left plaquette)
- Some contain 3, number 0→2 (as shown in figure, right plaquette)

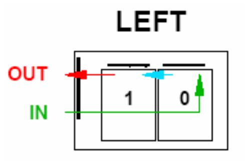
Plaquette Numbering Scheme



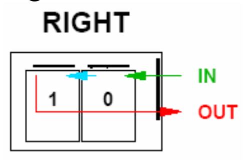
Readout Chips (within a plaquette)

- There are 7 types of plaquettes (VHDI) in total.
- The chip numbering scheme for the different types are as shown in the figures.
- The numbering scheme is the same as the order in which the token is passed.

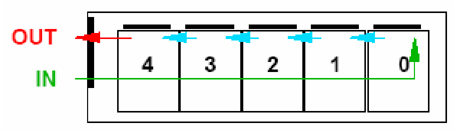
1. 1x2 plaquette with readout on the left



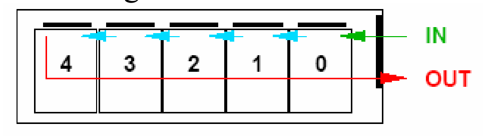
2. 1x2 plaquette with readout on the right



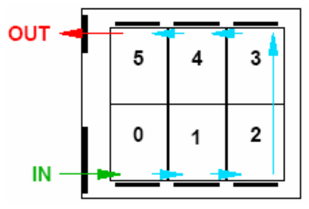
3. 1x5 plaquette with readout on the left



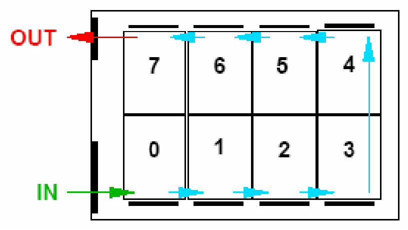
4. 1x5 plaquette with readout on the right



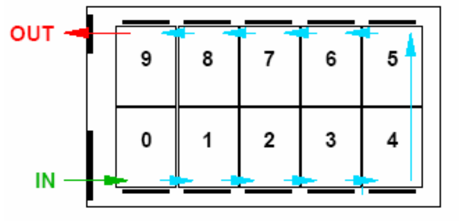
5. 2x3 plaquette



6. 2x4 plaquette



7. 2x5 plaquette



Pig Tail Connector Card

- The HDI pig tail of each panel enables data and control signals transfer to and from the plaquettes (chips).
- Pig tails of six adjacent panels (i.e. 3 blades) are connected to one connector card.
- The connector card is connected to a Port Card containing electro-optical converters, power, etc. on the other side.
- Bias voltage distribution and filtering goes through the Connector Card as well.
- The Connector Card basically contains a fan in (6:1) and fan out (1:6) circuits.

Port Card

Databases

CMS has defined 4 databases

1. **Construction Database**
2. **Equipment Management Database**
3. **Configuration Database**
4. **Conditions Database**

Pixel Construction Database Pixel

Construction Database Information

- We need a **Construction Database** to keep track of every component and its history during construction. The construction database will contain information on all components used in building a module. This includes results of various tests conducted on the components and QC parameters used to select the components. The Construction Database will be used for component selection during the module building process.

- In the case of the pixel detector, the construction database should start at the wafer level and end at the ladder (for barrel) or panel (for forward).

- The general assumption is that the online databases will keep track of only active components. There is no need for the database to keep track of individual passive elements like the Beryllium panel, silicon base plates (HDI), etc. besides noting their existence and the amount of material they contribute.

- History tables: Each table will have a history table associated with it to be able to keep track of any attribute change. The table itself will contain the latest set of attribute values. Old attribute values can be retrieved from the associated history table. This table will be used to track the history of any attribute of an entry (row) in the table.

- Associated with each table will be the following four attributes

Create Date	- datatype: date Updated when a new database entry is made (INSERT statement)
Create User	- datatype: varchar2 (name of user) Updated when a new database entry is made (INSERT statement)
Modify Date	- datatype: date Updated when an already entered attribute is altered with an UPDATE statement
Modify User	- datatype: varchar2 (name of user) Updated when an already entered attribute is altered with an UPDATE statement.

- When an UPDATE command is issued, the entry (row) targeted for modification is copied in its entirety (all attributes) to the associated history table prior to the actual modification. Only after the copy has completed will the modification process commence.

- There will be a wafer table for each wafer type - e.g. ROC, sensor, TBM, etc. - that will contain various attributes that describe the wafer.

- An Oracle record is a variable type. Memory is allocated only when data is entered. When an attribute contains a NULL value, no memory is used.

- Tables in need of a unique attribute (whose value is never repeated) will have an attribute that will contain a sequence number generated by Oracle. This attribute is updated (by Oracle) whenever a new entry is made.

- **The ROC_MAP table can be assumed to be the starting point for module or plaquette construction**. This is the stage during which a set of chips with compatible electrical properties are selected (16 or 8 + 2 or 1 extras) for module construction. Mating with a compatible sensor is done in the Module table.

- Since **the ROC_MAP** table is simply a mapping table, it **will not have a history table associated with it**.

- Since several institutions are involved in the fabrication process, having a central database that is very reliable and highly available with secure remote access will be necessary.

- It is also quite plausible that we may want the version numbers of test software stored in the database.

General Questions about the Pixel Construction Table

The Construction Database will contain information about every component used in constructing the detector:

- **Sensors**
- **Readout chips**
- **Electronic components (VHDI, HDI, FEC, FED, TBM, etc)**
- **Cables**
- **Cooling**
- **Mechanical structures**
- **Etc.**

For example information about a readout chip (ROC) could contain the following information:

- **Wafer serial number:** serial number of the wafer the chip was diced from
- **Sensor ID:** ID of the sensor bonded to the readout chip
- **ROC Test parameters:** results of various QC tests conducted on the chip for selection
- **Dicing, bump bonding, wire bonding, etc. information**
- **Module number:** ID of the module containing the ROC
- **Detector location:** which detector is the ROC in – (barrel, shell, layer), (disk, side, layer)?
- **ROC DAC settings (**initial?**)**
- Etc.

It is also quite plausible that we may want various software versions also stored in the database. Software is as critical to data taking as hardware. Should errors in software result in data taking problems that will have to be investigated. Having available versions of software used in data-taking will help troubleshooting.

Other entries in the Construction Database may include those related to

- **Sensors**
- **Modules**
- **Electronics**
- **Cooling**
- **Etc.**

Should the Construction Database end at the module level?
Will the following components have unique identifiers (for example, from manufacturer):

Wafer: Yes
ROC: No
Sensor: No (yes)
HDI: Yes
VHDI: Yes (No?)

What test parameters should we input in the various tables?

Items that may need to go into the Construction Database include
Be plate related info: No
Glue related info: No
Cooling loop: one loop for 6 panels: Yes
    Temperature monitor: @ input and @output
    Humidity monitor for half-cylinder???
    Pressure drop monitor: where???

Detector Table Definitions

General Information Regarding Tables

- Every entry in a table will have the following attributes:

Create_date	Date the entry was created. It is inserted when a new database entry is made.
Create_user	User responsible for making the entry. It is inserted when a New database entry is made.
Modify_date	Date the entry was last modified. It is updated whenever an existing database entry is altered.
Modify_user	User responsible for modifying an existing database entry. It is updated whenever an existing database entry is altered.

- Tables in need of a unique attribute whose value is never repeated will have an attribute that will contain a sequence number generated by Oracle. This attribute is updated (by Oracle) whenever a new entry is made.

- History tables: Each table will have a history table associated with it to be able to keep track of any attribute change.

- The table itself will contain the current (up to date) set of attribute values. Older attribute values can be retrieved from the associated history table.

Wafer, ROC, and Sensor Related Information

- There will be 3 different wafer tables for different wafer types
    - ROC wafer
    - Sensor wafer
    - The rest (TBM, Hub, etc.) wafer

- Each wafer has a unique ID allocated at time of manufacture.

- A wafer table will contain the various attributes (columns) that describe the wafer.

- Each wafer table has associated with it a history table.

- Meaningful tests of individual pixels can be made only during wafer tests and after module assembly.

- To track the behaviour of individual pixels, a selected number of identical tests must be performed during the two **stages**, *onwafer* and after *integration*.

- **Original plan:** Have a single wafer table that contains information of all wafers – readout chips (ROC), sensors, token bit manager (TBM), communication network hub (CNH), and silicon base plates.
- **Present plan**: This has been changed to have different wafer table for ROCs, sensors, and the rest (TBM/HUB/etc). There will be 3 wafer tables.

- (Do we need to store information on silicon base plates?) – **Answer: No**.

Readout Chip Wafer & Sensor Wafer

- The numbering (layout) of chips in a ROC wafer should be standard, i.e. the same in all cases (barrels or disks). A possible scheme is one used by Beat Meier.

- A PSI46 wafer (as tested in Fermilab) contains 66 reticules, and 4 chips per reticule.

- Before being diced, a wafer undergoes a series of tests and measurements.

- Tests include (as presented by Roland Horrisberger on March 16, 2004)
    - IV curve of power consumption (digital & analog)
    - Checking all 4160 pixel for readout by internal calibrate pulse
    - Checking of all data buffers in each double column
    - Checking of all timestamp buffers in each double column
    - Checking of overflow resets
    - Scanning for individual pixel threshold by varying cal pulses
    - Check for functioning mask bit of all pixels
    - Check for functioning trim bits (4bit) of all pixels by determining all trim bit shifted pixel thresholds by cal pulses

- Results of wafer tests conducted at PSI and Fermilab will be recorded in the wafer and corresponding component (ROC, sensor, etc.) tables.

- Results of sensor wafer tests conducted at PSI and Purdue will be recorded in the wafer and Sensor tables

- **Results from these tests will be used to categorize and select chips (ROC wafer). A chip has a definite placement in a wafer. How is this chip tracked after the dicing process? This becomes quite crucial since chips are selected based on these test results to construct a plaquette (disks) or a module (barrel). In the case of the barrel, this is not a problem since the various steps - e.g. bump deposition, bonding, etc. - are done in-house. However, in the case of the Forward Disks , this could prove very difficult since these steps are all going to be done at vendor sites. It is very important that methods are developed to track the individual chips.**

- **Results from these tests will be used to categorize and select sensors (Sensor wafer).**

- The next set of tests will be after dicing and bump bonding the chips to the sensors and constructing modules and plaquettes (in the case of the barrel, these tests can be performed only after a module has been constructed; in the case of the Forward Disks, these tests are performed with plaquettes.).

- **We need to know the quality of the chips (& sensors) being mounted on a plaquette/module. That means we need to have a way of tracking the chips (& sensors) being used.**

- It is very likely that the results of the tests conducted (as part of a wafer and as part of a plaquette/module, i.e. with and without a bonded sensor) will be different. We should save both sets of measurements.

- **Original idea:** It may be necessary to have a separate table to store test results of the individual on-wafer chips & sensors (the other table contains results of tests of individual chips bonded to sensors).
- **Current idea:** Both sets of individual ROC (Sensor) attributes will be stored in a single ROC (Sensor) table.

- Question: Do we need to correlate test data of a chip when it is still on a wafer with that after the chip is bonded to a sensor and connected to a plaquette (module)?

- Question: Will data from wafer tests be used for categorizing and selecting chips? Answer: Yes.

IV & CV Curves

IV-curves

Gino: Hopefully we can get a measurement on wafer from the manufacturer

- Before bump deposition (This is onwafer done at Purdue in our case)
- After "final" integration (This is for the plaquettes testing so I would say
  - Before burn-in
  - After burn-in
- After mounting on the stave (Blade for Forward)

Tilman: There are parameters measured using test structures on the wafer

- CV of one (or more) diode(s)
- All kinds of technical in formation:
  - Sheet resistance
  - "punch through voltage"
  - V_{FB} – flat-band voltage

Monitoring Information

- Associated temperature (at which the measurements were made) (this should be recorded for every measurement)
- Associated humidity (at which the measurements were made) (this should be recorded for every measurement)

Questions

- **What parameters are to be used for QC selection? How does one select the grade of the ROC?**

Correlating ROC and Sensor information

- **When chips are bonded to sensors, information from the on-wafer ROC table and on-wafer sensor table need to be mated (brought together).**
- What parameters do we use to bond a sensor to a set of ROCs?

<u>Explanation of ROC testing (from Danek)</u>

ROC wafer

- This test includes testing each pixel: response, threshold, trim scan
- It also has some global ROC tests like DAC programming and digital/analog current measurements
- At the end a ROC will be characterized by the percentage of BAD pixels
- several limits:
    - best ROCs - selected layers, e.g. barrel at 4cm
    - good ROCs - other layers e.g. 11cm barrel
    - lab/reserve ROCs
    - useless ROCs to throw away

The wafer table should have a list of ROC quality in some reasonable bins.
Sensor wafers will also be tested before sensors are bumped (IV tests, ...)

Diced/bumped ROCs ???

- Visual inspection (using a microscope) for damage and alignment. In the past it has been observed that sometimes after bump-bonding 1 ROC is tilted/rotated resulting in many failed pixel bonds, one can clearly see this visually.
- Some electrical tests can be done on selected bumped modules ("bare modules") with a probe station. This is time consuming and very delicate work so we hope to avoid doing it for each bare module.

Barrel HDI's

- Bare barrel HDIs will be first tested for simple shorts (probably done by the producer).
- Shorted HDIs will be thrown away.
- After the TBM + cables + any other chips/components are mounted on the HDI, the whole assembly will be fully tested.
- This will include: clock distribution, I2C signal distribution, power, token out, trigger, etc.
- Only fully functional HDI assemblies will be glued to the bare modules.

Full Module Testing

- Module testing will be the most complete procedure.

- Repeat the ROC tests from the wafer testing but also extend them.
- For example
    - Do full calibration of each pixel (gain, offset) of each pixel.
    - Do Threshold and noise scans.
    - Cycle the temperature and repeat some tests after.
- For Atlas pixels testing of 1 module takes about 8 hours. Some modules (maybe not all) will be also tested with sources.
- Scan DAC settings, test the bias voltage etc.
- Finally the modules will be classified into several groups according to the quality.

TBM, HUB, etc.

Again I am not sure how much testing we can do with the TBM after the wafer testing and before they are mounted on modules? Cables, Hubs etc. should be tested.

ROC_WAFER Table

Table Name: ROC_WAFER

Table Description

This table will contain information about individual ROC wafers. It has associated with it a history table that will record every entry along with all changes to exiting entries in the table. The updates are automatically carried out using “triggers”.

Attributes

Upon delivery of wafer, the following parameters are noted

• **rocwaf_id**	**Primary key**. Unique ID of wafer given by manufacturer
• **manf_name**	Name of manufacturer
• **manf_date**	Date of manufacture
• **batch_id**	Batch number (assuming wafers are received in batches)
• **rel_id**	Relative wafer number (number within batch)

Note: The (batch_id, rel_id) combination can be used as a “working number” of a wafer. Both of these numbers are prescribed upon reception (by the end user, i.e. us) and hence are relative. The combination maps to the **rocwaf_id** attribute.

• **create_date**	date updated when a new entry is made in the database
• **create_user**	username updated when a new entry is made in the database
• **modify_date**	date updated when an existing attribute is altered
• **modify_user**	username updated when an existing attribute of an entry is altered

Conduct a visual inspection of the wafer (for scratches, damages, etc.). Take a photograph if faults detected (wafer quality)

• **vis_insp**	Result of visual inspection. Will contain a picture if scratches or damages exist.

Test every chip in the wafer. The test date is entered in the wafer table. However, the results from the test of the individual on-wafer chips are entered in the ROC table with the **stage** attribute set to ***onwafer***.

• **test_date**	Date the wafer is tested
• **thickness**	thickness of ROC wafer

Wafer is sent for thinning. Upon reception, wafer is visually inspected and the following recorded

• **grind_vendor**	name of vendor responsible for grinding
• **grind_date**	date of grinding
• **stress_test**	result of stress test: Y or N
• **grind_vis_insp**	visual inspection after grinding. If there are problems, this will contain a picture.

Wafer is sent for bump deposition. Upon reception, the wafer is visually inspected and the following recorded.

• **photres_exp_date**	photo resist expiration date
• **sputter_setting**	Setting used for bump deposition
• **bump_thickness**	Thickness of bump deposition
• **missing_bumps**	List of chips with missing bumps
• **merged_bumps**	List of chips with merged bumps
• **defective_bumps**	List of chips with defective bumps

This table contains a "list of chips" with bump problems. The "list of pixels" with bump problems in a ROC (detected at this stage) will be in the ROC table and the **stage** attribute value set to ***onwafer***.

The wafer is sent for dicing. Upon reception, the individual chips are visually inspected for damage. Information from visual inspection of individual chips is entered in the ROC table.

ROC_WAFER_HISTORY Table

Table Name: **ROC_WAFER_HISTORY**

Table Description: The ROC_WAFER_HISTORY table will contain the rocwaf_id as the foreign key, all the attributes of that wafer (old and new), and a comment

Attributes:

• **rocwaf_hist_id**	Oracle generated sequence number, primary key
• **rocwaf_id**	(**foreign key**) primary key in ROC_WAFER table
• **rocwaf_timestamp**	should have time with time zone, unique key
• **rocwaf_hist_comment**	comment

- **All attributes** of ROC_WAFER

Comments

- A table entry (row) needs an attribute that takes on a value that is unique, i.e. the attribute never takes on the same value more than once so that it can be distinctly identified.
- A wafer history table can have several entries (rows) for the same wafer for changes made at different times.
- It is possible that the **rocwaf_id**, together with the **rocwaf_timestamp** attribute, could constitute a unique ID. A unique attribute that maps to these multiple attributes helps simplify situations.
- **rocwaf_hist_id** is an Oracle generated sequence number that serves this very purpose. A number is never repeated in this sequence and hence maintains a unique ID for each entry in the table.
- The history table will be populated on updates (via triggers) with whatever values (columns) for the **rocwaf_id** there are in the **ROC_WAFER** table (in addition to the other attributes defined in the table).

Comments (Oracle Designer Related)

- There is no need to specifically enter a Foreign Key attribute in a table. The relationship should automatically update a table with a foreign key from another table.

- Draw a one-to-many (1:M) from the roc_wafer table to the roc_wafer_history table. For each wafer entry, there can be several entries in the history table.

- Make the relationship M:M, i.e. mandatory at the ROC_WAFER and ROC_WAFER_HISTORY ends. Mandatory because for every entry/change in the ROC_WAFER table there will be an entry in the history table. A table relationship becomes mandatory when the primary key of the table cannot take on a NULL value. The primary key of ROC_WAFER, rocwaf_id cannot take on a NULL value, and the primary key of the ROC_WAFER_HISTORY, rocwaf_hist_id, cannot take on a NULL value either. Hence the relationship is a mandatory to mandatory.

- Use the "Edit Relationship" option (by clicking on the relation in the diagram) and check the "UID" box in the "to roc_waf_hist" end if the foreign key in a table is also a primary key. In this case the ROC_WAFER_HISTORY table. This causes a "crossbar" to be drawn at the 'multiple' end of the relationship foreign key is not the primary key in the ROC_WAFER_HISTORY table. This causes a "crossbar" to be drawn at the 'multiple' end of the relationship.

- The ROC_WAFER table is the parent and the ROC_WAFER_HISTORY table is a child.

- The primary key, rocwaf_id, in the ROC_WAFER table becomes a foreign key in the ROC_WAFER_HISTORY table.

- When a primary key in a given table is to be a foreign key in another table, the primary key does not need to be specified in the table in which it is supposed to be a foreign key. The foreign key is automatically generated by the relationship. In the case of ROC_WAFER_HISTORY table, the foreign key is rocwaf_id. This is the primary key in the ROC_WAFER table and does not have to be specified in the ROC_WAFER_HISTORY table. It is introduced as a foreign key by the relationship during the data transformation (DDT) process.

- Attribute sequence numbers start at 10 with an interval of 10. This is to not have to redo the sequence numbers when attributes are inserted in between.
- Create the following domains

○ create date	type: date
○ create_user	type: varchar2
○ modify_date	type: date

    - modify_user type: varchar2
- Use the Attribute Details option to set each of the attribute types to the corresponding domain types where applicable.

Questions?

- **What additional attributes do we need to add to the ROC_WAFER table to incorporate the various procedures the wafer goes through, e.g. thinning, etc.?**

SENSOR_WAFER Table

Table Name: SENSOR_WAFER

Table Description

This table will contain the various parameters for sensor wafers before they are diced.

Attributes

• **senswaf_id**	primary key
• **manf_name**	
• **manf_date**	
• **batch_id**	
• **rel_id**	Relative wafer number (number within batch)
• **create_date**	date updated upon a new database entry
• **create_user**	username updated upon a new database entry
• **modify_date**	date updated when an existing attribute is altered
• **modify_user**	username updated when an existing attribute is altered

Conduct a visual inspection of the wafer (for scratches, damages, etc.). Take a photograph if faults detected (wafer quality). This is done before wafer is sent for bump deposition and dicing.

• **vis_insp**	Result of visual inspection. Will contain a picture if scratches or damages exist.

Test every sensor in the wafer. The test date is entered in the wafer table. However, the results from the test of the individual on-wafer chips are entered in the Sensor table. However, some parameters like sheet resistance (Ohm/square), punch through voltage, and flat-band voltage are entered in the Sensor Wafer table.

• **test_date**	Date the wafer is tested
• **thickness**	thickness of the sensor wafer
• **planarity**	(bow in the wafer)
• **sheet_resist**	sheet resistance (on test structure). Must be

	$2\ k\Omega cm < \rho < 5\ k\Omega cm$
• **punch_volt**	punch through voltage (on test structure). Must be > 3V
• **FBvolt**	flat-band voltage

CV curve measurements are made using test structures in the wafer. IV curves are also generated for all sensors, but the results are entered in the Sensor table.

• **CV**	CV curve of test structure in sensor wafer. Data type will be BLOB. This is measured on various test structures on the wafer???
• **tempCV**	Temperature at which CV curve is measured
• **humidCV**	Humidity during CV curve measurement
• **Vdepl**	Depletion voltage calculated from CV curve (test structure)
• **Grk_Test**	Passed “Greek Cross” test – Yes/No. Contacts in a special test structure must be open

Wafer is sent for dicing. Upon reception, the individual sensors are visually inspected for damage. Information from visual inspection of individual chips is entered in the Sensor table.

Questions?

- **What additional attributes do we need to add to the SENSOR_WAFER table to incorporate the various procedures the wafer goes through, e.g. dicing, thinning, etc.?**

SENSOR_WAFER_HISTORY Table

Table Name: SENSOR_WAFER_HISTORY

Table Description

This table will contain the various parameters for sensor wafers before they are diced.

Attributes

• **senswaf_hist_id**	(**primary key**) Oracle generated sequence number,
• **senswaf_id**	(**foreign key)** - primary key in SENSOR_WAFER table
• **senswaf_timestamp**	time with time zone, unique key
• **senswaf_hist_comment**	comments

- **All attributes** of the wafer pointed to by senswaf_id

TBM_WAFER Table

TBM_WAFER_HISTORY Table

ROC (readout chip) Table

Table Name: ROC

Table Description:

There will be only one ROC (readout chip) table that will contain information on all individual ROCs from various stages.

The attributes (parameters) that describe a ROC can be divided into two main groups – *onwafer* attributes resulting from on-wafer measurements (before dicing) and *integrated* attributes resulting from measurements after the ROCs have been integrated (after dicing, bump-bonding, etc.) onto modules/plaquettes.

Associated with each individual ROC are entries (rows) in the table which will contain all the attributes (columns) that describe it. Currently, the table will have two entries per ROC, one for chip measurements made before the wafer is diced (*onwafer* measurements) and the second after the chip is integrated onto a module (*integrated* measurements).

For a given ROC, the **stage** attribute which can take two values, *onwafer* or *integrated*, will determine which test phase the attributes belong to – since those prior to dicing are expected to be quite different from those after the chip is bonded. A unique entry in the table will be defined by a combination of

rocwaf_id of the wafer the chip is derived from
roc_id of the chip within the wafer (assuming a fixed ROC numbering scheme, e.g. as that used by Beat Meier)
stage at which the attribute values were collected

An entry in the ROC table will also be identified uniquely by an Oracle generated sequence number, **rocseq_id**, which will serve as the primary key. In other words, the unique ID, **rocseq_id**, essentially maps to a combination of the three attributes described above. Every entry in the ROC table will have a unique **rocseq_id** – even if they correspond to the same physical chip. If it so happens that an attribute is valid only for one stage (e.g. **onwafer**), that attribute will take on a NULL value for the other (e.g. **integrated**).

Attributes

• **rocseq_id**	Oracle generated sequential number, **primary key**
• **rocwaf_id**	**foreign key** from ROC_WAFER table
• **roc_id**	relative number of the roc within the wafer, **rocwaf_id & roc_id** together will constitute a unique key for a ROC
• **stage**	Stage at which measurements are made: *onwafer* or *integrated*.
• **detector**	detector type associated with sensor (barrel/disk)
• **create_date**	type: date
• **create_user**	type: varchar2
• **modify_date**	type: date
• **modify_user**	type: varchar2

The following attributes are generated after bump deposition on the wafer. Technically they will come under the *onwafer* stage.

• **missing_bumps**	List of missing bumps in chip
• **merged_bumps**	List of merged bumps in chip
• **defective_bumps**	List of defective bumps in chip

The following attribute is generated after dicing of a dumped chip. It will be grouped in the *onwafer* stage.

• **visual_insp**	visual inspection of chip after dicing. If chip is damaged, it will contain a picture of the chip
• **roc_grade**	Grade of ROC determined from onwafer tests. This attribute could be used for chip selection.

Supply voltages and currents

• **Sup_Vana**	supply analog voltage
• **Sup_Iana**	current drawn by ROC from analog supply
• **Sup_Vdig**	supply digital voltage
• **Sup_Idig**	current drawn by ROC from digital supply

The IV curves will contain a collection of (voltage, current) number pairs. If necessary, a user interface can present it graphically.

• **Curve_IVana**	IV curve for analog voltage
• **Curve_IVdig**	IV curve for digital voltage

Check response of every pixel using internal calibrate pulse and determine which pixels are bad and/or which double columns are bad. Bad → unusable. This will have to be done during both stages.

• **bad_pixels**	list of all bad pixels
• **bad_dcols**	list of all bad double columns

Scan pixel threshold (vary calibrate pulse) and determine the thresholds. This will have to be done during both stages.

• **pixel_thrs**	thresholds of all individual pixels

Scan pixel trim bits to see if they function. This will have to be done during both stages.

• **bad_trimbits**	list of pixels with defective trim bits (type:varchar2)

Check if the mask bits of the individual pixels function. This will have to be done during both stages.

• **bad_maskbits**	list of pixels with defective mask bits (type:varchar2)

For each double column, check if the double column data buffers and time-stamp buffers function. This will have to be done during both stages.

• **bad_buffers**	list of bad buffers (32 max, type: varchar2).
• **bad_tsbuffers**	list of bad time-stamp buffers (12 max, type: varchar2).

It is assumed that the attributes **bad_trimbits** and **bad_maskbits** will be stored as variable character type (varchar2). In the case of the **bad_maskbits**, the list will only contain the pixel numbers; whereas in the case of the trimbits, it may be possible to have a pixel number and the associated defective bits.

In the case of the bad_buffers, the list will contain the double column number and the numbers of the associated buffers that are assumed to be bad. The same for bad timestamp buffers.

The following attributes represent the DAC settings of a ROC. The DAC settings of a ROC during the two stages, **onwafer** and **integrated**, will be very different. The relevant values, however, are those corresponding to the integrated stage. These are the values that will be downloaded on to the chip and are likely to vary with time.

Tracking DAC settings in time: Whenever a new DAC setting is entered in the database (an UPDATE operation), an Oracle trigger will copy the entire existing ROC record to the associated history file and then only proceed to make the necessary change(s) to the ROC entry. The time behavior of a given DAC of a particular ROC can be obtained by querying the history file for all entries corresponding to that ROC.

It is also assumed that the database will contain DAC response curves for each integrated ROC. The *onwafer* ROC attributes will have NULL values for these attributes. Details for obtaining the response curves have not been specified.

Method to obtain DAC response curves may be as follows (Wolfram):

1. The inter-relation of DACs themselves is not very strong and will be even weaker for the new chip.
2. Use a standard setting for the other DACs when one is tested. These (standard) values will have to be determined at a later date.

Supply DACs

• **Vana**	analog power
• **Resp_Vana**	Response curve for Vana
• **Vsh**	sample & hold power
• **Resp_Vsh**	Response curve for Vsh
• **Vcomp**	comparator power
• **Resp_Vcomp**	Response curve for Vcomp
• **Vdig**	digital power
• **Resp_Vdig**	Response curve for Vdig

Analog PUC

• **Vleak_comp**	leakage current compensation
• **Resp_Vleak_comp**	Response curve for Vleak_comp
• **VrgPr**	preamp feedback
• **Resp_VrgPr**	Response curve for VrgPr
•	
• **VwllPr**	preamp fb-well-voltage
• **Resp_VwllPr**	Response curve for VwllPr
• **VrgSh**	shaper feedback
• **Resp_VrgSh**	Response curve for VrgSh
• **VwlSh**	shaper fb-well-voltage
• **Resp_VwlSh**	Response curve for VwlSh
• **VhldDe**l	Sample & Hold delay
• **Resp_VhldDel**	Response curve for VhldDel
• **Vtrim**	pixel trim range
• **Resp_Vtrim**	Response curve for Vtrim
• **VthrComp**	pixel comp threshold
• **Resp_VthrComp**	Response curve for VthrComp
• **Vcomp_casc**	pixel comp cascade
• **Resp_Vcomp_casc**	Response curve for Vcomp_casc

Pixel Readout

• **VIbias_Bus**	DC-readout bias current
• **Resp_VIbias_Bus**	Response curve for VIbias_Bus
• **Vbias_sf**	pixel to DB sf-current
• **Resp_Vbias_sf**	Response curve for Vbias_sf

Double Column Readout

Data buffer read-amplifier

• **VoffsetOP**	Offset voltage
• **Resp_VoffsetOP**	Response curve for VoffsetOP
• **VIbiasOP**	On current
• **Resp_VIbiasOP**	Response curve for VIbiasOP
• **VIbiasOff**	Standby current
• **Resp_VIbiasOff**	Response curve for VIbiasOff

DC-readout buffer amplifier

• **VoffsetRO**	offset voltage
• **Resp_VoffsetRO**	Response curve for VoffsetRO
• **VIon**	On current
• **Resp_VIon**	Response curve for VIon
• **VIoff**	Standby current
• **Resp_VIoff**	Response curve for VIoff

Chip Readout

Pulse height diff. amplifier

• **VIbias_PH**	amplifier bias current
• **Resp_VIbias_PH**	Response curve for VIbias_PH

DAC (event-multiplexer)

• **Ibias_DAC**	DAC gain current
• **Resp_Ibias_DAC**	Response curve for Ibias_DAC

Last DAC amplifier

• **Ibias_Idac**	Bias current
• **Resp_Ibias_Idac**	Response curve for Ibias_Idac

Chip readout amplifier

• **VIbias_roc**	bias current
• **Resp_VIbias_roc**	Response curve for VIbias_roc

- **CalDel**
- **RangeTemp**

A BLOB attribute to store pixel number (row, col), trim bits (4), and mask bit (1) for all of 4160 pixels (80 rows x 52 columns)

- **pixelinfo**

It is also very likely that after a ROC has been integrated into a module/plaquette, a response curve for each individual pixel is determined. A fit to the Vcal response curve is obtained with a few parameters (~6 or so). Save these fit parameters as attributes.

- **Vcal_fit_0**
- **Vcal_fit_1**
- **Vcal_fit_2**
- **Vcal_fit_3**
- **Vcal_fit_4**
- **Vcal_fit_5**

Average analog levels (ADC counts): It may be necessary to have the analog levels (in ADC counts and the width of distribution) in the ROC table.

• **analog_c0**	Least significant column
• **width_analog_c0**	width
• **analog_c1**	Most significant column
• **wdith_analog_c1**	width
• **analog_r0**	Least significant row
• **width_analog_r0**	width
• **analog_r1**	Next to most significant row
• **width_analog_r1**	width
• **analog_r2**	Most significant row
• **width_analog_r2**	width
• **analog_UB**	Ultra Black reference level

The following do not go into the database

Fast Trigger (Do these have to be in the database???)

- roc_VIColOr
- roc_Vnpix
- roc_VsumCol

Questions:

- **Can a list of bad double columns be stored in a single column? Yes. Can be in principle stored as a character string, e.g. VARCHAR2)**
- **Can a list of bad pixels be stored in a double column? (Yes, as a VARCHAR2 string as mentioned above)**

- **How should the BLOBs be formatted?**

- **Is it possible to simply copy a table and convert it into a history table? NO.**
- **For example, can the ROC table be copied to a new table and name it ROC_HISTORY table and then add the additional attributes? In short, how can one get around having to type in a large number of attributes that already exist in another table? No, cannot. Every attribute needs to be typed in.**

- **Do we need to measure IV curves for every ROC? If so, what voltage granularity do we use? Maybe**

- **Should we store the Fast Trigger attributes in the database?**
- **Should we keep track of temperature and humidity measurements?**

ROC Table Comments:

- The attribute combination of **rocwaf_id**, **roc_id,** and **stage** enables the identification of a particular chip.

- For a given readout chip, the attribute **stage** helps distinguish the parameters measured before the wafer was bumped and diced from those after the chip was bumped, diced, and integrated onto a module.

- The primary key (unique and non-NULL ID) is the Oracle generated sequential ID, **rocseq_id**. This is required because the only other way to get a unique ID of a database entry is to use a combination of
    - **rocwaf_id**: (a foreign key)
    - **roc_id**: the relative ROC number within the wafer
    - **stage:** the stage when the measurements were made

  Having a single primary key is important for mapping purposes.

- The **detector** attribute specifies the detector, barrel or forward, the chip resides in.

- When a new ROC entry is made in the table, Oracle automatically "fires" a trigger to track the date and time of entry (**create_date**) and the ID of the initiating user (**create_user**). Similarly when an existing entry is modified (info added, deleted, or changed), a corresponding trigger updates the **modify_date** and **modify_user** attributes.

- A ROC is initially graded (attribute: **roc_grade**) based on tests and measurements conducted on the ROC while on the wafer (before bumping and dicing). How exactly the ROCs are graded has yet to be decided. ROCs will be selected for integration based on the value of this attribute.

- The following attributes represent the supply voltages (analog and digital) at which the ROC is operated together with the currents drawn.
    - **Sup_Vana**
    - **Sup_Iana**
    - **Sup_Vdig**
    - **Sup_Idig**

- The attributes **Curve_IVana** and **Curve_IVdig** represent the analog and digital IV curves for the ROC.

- The double columns of a ROC that are designated "bad" will be listed (as characters) in the **bad_columns** attribute while individual pixels that are designated "bad" will be listed (as characters) in the **bad_pixels** attribute.

- The DAC settings for a ROC will definitely be different while on-wafer (before dicing) from those after integration (after dicing, bump-bonding, etc.). It is conceivable (but not certain) that the DAC settings for all ROCs on a given wafer may be the same. It is unlikely that this set of attributes may need to be altered at all.

- However, the DAC settings after integration (**stage**: ***integrated***) will need to be changed with time. This set of attributes will contain the most up-to-date values to be downloaded onto the chips for operation. These individual ROC settings will probably come into the picture when the chips are tuned during module/plaquette testing.

- There are two sub-groups of attributes that are used to record the average value of the various address (row and column) levels (in ADC counts) together with the widths of the distributions.

○ **analog_c0**	(least significant column)
○ **width_analog_c0**	
○ **analog_c1**	
○ **wdith_analog_c1**	(most significant column)
○ **analog_r0**	(least significant row)
○ **width_analog_r0**	
○ **analog_r1**	
○ **width_analog_r1**	
○ **analog_r2**	
○ **width_analog_r2**	(most significant row)

- These values could be used for tracking signal levels of the analog levels of the ROC. Two attributes have also been defined to track the analog level of the “Ultra Black” level of the ROC before and after integration. **Are these attributes needed**?

- What information should we gather during burn-in? We need to define the burn-in process properly.

- We need to start defining the calibration procedure (process) so that we have a fairly good idea of parameters that should go in the DB.

- Upon an UPDATE command of the ROC table, all attributes (columns) of the ROC (row), that is being altered, are first copied to the ROC_HISTORY table. This is initiated by an Oracle trigger.

- Only after the copy has been completed are the changes implemented.

- This enables a user to scan a history table and track a ROC attribute(s) through time.

Conditions Information (we may have to get this information during construction)

- Gain (computed from calibration run Vcal)
- Pedestal (computed from calibration run)
- Threshold
- Noise
- Efficiency
- Average rate
- Offset

ROC_HISTORY Table

Table Name: ROC_HISTORY

Table Description:

There will be one table, **ROC_ HISTORY** table, which will document any change made to the **ROC** table. The documentation process will be triggered by either an *INSERT* or *UPDATE* operation performed on the ROC table.

Attributes

• **roc_hist_id**	This is an Oracle generated sequence number which will give the entry (row) a unique ID. It will also serve as the **primary key** of the table.
• **rocseq_id**	The ROC sequence ID is a **foreign key**
• **roc_hist_timestamp**	time of history record registration. It should have time with the time zone included. It, together with the **rocwaf_id**, can constitute a unique key.
• **roc_hist_comment**	Comments (user input)
• **all ROC attributes**	

Comments:

- The unique sequence number, **roc_hist_id**, is essential since a particular chip could have several entries in the history table.

- Upon an insert or update (triggered by an Oracle trigger), Oracle copies all the attributes for the readout chip, rocseq_id, being altered to the history table before making the changes. In addition, the history table will also contain the other attributes defined in the table.

SENSOR Table

Table Name: SENSOR

Table Description:

There is one **SENSOR** table that will contain information on all individual sensors. Each sensor will have entries to record information gathered from various stages. Just as in the case of the ROCs, measurements for each sensor are carried out in two stages, *onwafer* and *integrated*. That means there will be two entries for each sensor.

Just as in the case of the ROCs, the attributes (parameters) that describe a sensor can be divided into two main groups – *onwafer* attributes resulting from on-wafer measurements (before dicing) and *integrated* attributes resulting from measurements after the sensors have been integrated (after dicing, bump-bonding, etc.) onto modules/plaquettes.

Associated with each individual sensor are entries (rows) in the table which will contain all the attributes (columns) that describe it. Currently, the table will have two entries per sensor, one for measurements made before the wafer is diced (*onwafer* measurements) and the second after the sensor is integrated onto a module (*integrated* measurements).

For a given sensor, the **stage** attribute which can take two values, *onwafer* or *integrated*, will determine which "stage" the attributes belong to – since those prior to dicing are expected to be quite different from those after the chip is bonded. A unique entry in the table will be defined by a combination of

senswaf_id of the wafer the chip is derived from
sens_id of the chip within the wafer (assuming a fixed ROC numbering scheme, e.g. as that used by Beat Meier)
stage

An entry in the SENSOR table will also be identified uniquely by an Oracle generated sequence number, **sensseq_id**, which will serve as the primary key. Every entry in the SENSOR table will have a unique **sensseq_id** – even if they are for the same chip. In other words, the unique ID, **sensseq_id**, essentially maps to a combination of the above three attributes the function described above.

For a given SENSOR, an attribute (column) can take different values depending on the observation **stage**. If it so happens that an attribute is valid only for one stage (e.g. ***onwafer***), that attribute will take on a NULL value for the other (e.g. ***integrated***).

Attributes:

There is one entry (row) per sensor.

• **sensseq_id**	Oracle generated sequential number, **primary key**
• **senswaf_id**	**foreign key** from sensor_wafer table
• **sens_id**	relative number of the sensor within the wafer, **senswaf_id & sens_id** together could constitute a primary key
• **detector**	detector type associated with sensor (barrel/disk)
• **create_date**	type: date
• **create_user**	type: varchar2
• **modify_date**	type: date
• **modify_user**	type: varchar2
• **stage**	Stage at which measurements are made: *onwafer* or *integrated.*

The sensor wafer is sent for bump deposition. Upon reception, the wafer is visually inspected and the following recorded. The following attributes are generated after bump deposition on the wafer. Technically they will come under the ***onwafer*** stage.

• **bump_thickness**	thickness of bump deposited on sensor
• **missing_bumps**	List of pixel with missing bumps
• **merged_bumps**	List of pixels with merged bumps
• **defective_bumps**	List of pixels with defective bumps

The following attribute is generated after dicing of a bumped sensor. It will be grouped in the *onwafer* stage.

• **Visroc_posm_insp** (before	visual inspection of sensor after dicing bonding). If sensor is damaged, it will contain a picture of the sensor
• **sens_type**	type of sensor depending on detector type
• **sens_grade**	grade of sensor determined from on-wafer tests. This attribute could be used for sensor selection.
• **sens_thickness**	Thickness of sensor

Attributes associated with wafer measurements (onwafer & integrated)

• **deplvolt**	full depletion voltage
• **brkvolt**	breakdown voltage
• **opvolt**	operation voltage
• **leakcur**	leakage current associated with operation voltage
• **Curve_IVsens**	IV curve measured for sensor (data type will be BLOB)
• **IVtemp**	temp at which measurements made
• **IVhumid**	humidity of environ during measurement

Comments:

Attribute Name: detector

- Describes the detector that the sensor is part of (barrel or forward)

detector: 0	→ Detector is barrel (module)
detector: 1	→ Detector is forward (plaquette)

Attribute Name: sens_type

Describes the type of sensor bonded to readout chips depending on the **detector** type.

If **sens_det** = 0, i.e. it is a module, the different sensor types possible are

sens_type: 0	F (full, 16 ROCs)
sens_type: 1	H (half, 8 ROCs).

If **sens_det** = 1, i.e. it is a plaquette, the different sensor types possible are

sens_type: 0	1x2	(L & R)
sens_type: 1	1x5	(L & R)
sens_type: 2	2x3	
sens_type: 3	2x4	
sens_type: 4	2x5.	

Attribute Name: sens_grade

Grade (or categorization) of a sensor

sens_grade: gold	**Gold:** $V_{Break} > 600V$ **and** $I_{Density} < 1nA/mm^2$
sens_grade: silver	**Silver:** $V_{Break} > 300V$ **and** $I_{Density} < 1nA/mm^2$
sens_grade: bronze	**Bronze:** $V_{Break} < 300V$ **and/or** $I_{Density} > 1nA/mm^2$
: bronze***	**Bronze***** = High guard ring current

Comments: (these figures are from Purdue for the forward pixels)

- Just as in the case of the ROCs, there are 2 entries for each sensor corresponding to the two different values of the **stage** attribute, *onwafer* and *integrated*.

- The primary key (unique, non-NULL ID) is the Oracle generated sequential ID, **sensseq_id**. This is required because the only other way to get such an ID is to use a combination of the sensor wafer ID (**senswaf_id**), a foreign key, and the relative sensor number within the wafer, **sens_id**. Having a single primary key is important for mapping purposes.

- When a new sensor entry is made in the table, Oracle automatically "fires" a trigger that records the date and time of entry (**create_date**) and the ID of the initiating user (**create_user**). Similarly when an existing entry is modified (update command issued when info is added, deleted, or changed), a corresponding trigger updates the **modify_date** and **modify_user** attributes.

- A sensor is graded based on tests and measurements conducted on the sensor while on the wafer. How this is done is yet undecided. Sensors (for bonding) will be selected for integration based on the value of this attribute.

- IV curves (set of numbers) will be only for a subset of sensors. Hence there will be an IV curve attribute, but it will be categorized as optional (i.e. can take on a NULL value). If there are IV curves from the manufacturer, do we create a separate attribute (column)? Should we have an IV curve attribute for each instance of measurement, e.g. before & after bonding, etc.?

  There will be an IV curve for every sensor!!!

- CV curve measurements are carried out on test structures. What table should they reside in – SENSOR WAFER table or SENSOR table? Ans. Sensor Wafer Table for now.

- What exactly are these parameters? When and how are they measured? Should they go in the SENSOR WAFER table or in the SENSOR table (with individual sensors)?

- What information should we gather during burn-in?

- The SENSOR table will have associated with it a history table

SENSOR_HISTORY Table

Table Name: SENSOR_HISTORY

Table Description:

There will be one table, **SENSOR_ HISTORY**, which will document any change made to the **SENSOR** table. The documentation process will be triggered by either an *INSERT* or *UPDATE* operation performed on the **SENSOR** table.

Attributes:

• **sens_hist_id**	This is an Oracle generated sequence number which will give the entry (row) a unique ID. It will also serve as the **primary key** of the table.
• **sensseq_id**	The sensor sequence ID is a **foreign key** in the history table
• **sens_hist_timestamp**	time of history record registration. It should have time with the time zone included. It together with the rocwaf_id can constitute a unique key.
• **sens_hist_comment**	Comments (user input)
• **all SENSOR attributes**	

Comments

- Upon an UPDATE command of the SENSOR table, all attributes (columns) of the sensor (row), that is being altered, are first copied to the SENSOR_HISTORY table. This is initiated by an Oracle trigger.
- Only after the copy has been completed are the changes implemented.
- This enables a user to scan a history table and track a sensor through time.
- The unique Oracle generated sequence number, **sens_hist_id**, is essential since a particular chip could have several entries in the history table. This attribute constitutes the primary key in the history table. A primary key attribute value should never be repeated, nor should it ever take a NULL value.

- The primary key in the SENSOR table, **sensseq_id**, is a foreign key in the SENSOR_HISTORY table.

TBM Table

Table Description

- This table contains information related to the individual TBMs.
- In the case of the barrel, each module in layers 0 and 1 contain 2 TBMs; while a module in layer 2 contains 1 TBM.
- In the case of the forward disks, each panel contains 1 TBM.

Attributes

Create Date	(updated when an entry is made in the database)
Create User	(updated when an entry is made in the database)
Modify Date	(updated when an already entered attribute is altered)
Modify User	(updated when an already entered attribute is altered)
Oracle Sequence Number	(Updated by Oracle after entries made in the database)
Wafer ID	
TBM ID	

Questions & Issues

- What parameters are used to define the behaviour of a TBM?
- What parameters are needed for quality selection purposes?

ROC_MAP Table

Table Name: ROC_MAP

Table Description:

Multiple ROCs are mounted on modules (barrel) and plaquettes (forward). The **ROC_MAP** table is a mapping table that will enable a rapid correlation of the modules/plaquettes with the mounted ROCs. This is the stage during which ROCs are selected for mounting on modules and plaquettes. The ROC_MAP table can be assumed to be the starting point for module construction. A set of chips with compatible electrical properties are selected (16 or 8 + 2 or 1 extras) for module construction. The selection process has yet to be defined – it could be carried out by hand or with the help of software.

There is one entry (row) for each individual ROC identified by **rocseq_id**. Each row (entry) will contain two unique attributes of which one is the **rocseq_id** and the other either **mod_id**, the module ID, or **plaq_id**, the unique plaquette ID, depending on which one the ROC is mounted on. The attribute **roc_pos**, the relative position of the ROC within the module (plaquette) is also stored in this table.

Attributes:

- **create_date**
- **create_user**
- **modify_date**
- **modify_user**

• **rocseq_id**	(**foreign key**) Oracle generated sequence number in the ROC table. It is unique and corresponds to a ROC from a particular **rocwaf_id** and **roc_id**.
• **mod_id**	(**foreign key**) The unique module ID
or	
• **plaq_id**	(**foreign key**) The unique plaquette ID, depending on which object the ROC is mounted on. They are both foreign keys in the MAP_ROC table.
• **roc_pos**	position of the ROC within a module or plaquette

Comments

- There will be multiple rows for a given **mod_id** or **plaq_id** since there are multiple readout chips in a module (plaquette).
- The **rocseq_id**, which is a primary key in the ROC table and a foreign key in the ROC_MAP table, is also the primary key in the latter table. Hence in the "Edit Relationship" option the "Primary UID" box in the ROC_MAP entity is checked.
- The ID of the plaquette, **plaq_id**, and the ID of the module, **mod_id**, are primary keys in the respective tables and foreign keys in the ROC_MAP table.
- The relationship from ROC to ROC_MAP is mandatory.
- **For a given rocseq_id, there will be a one to one relationship between ROC and ROC_MAP tables. Although the relationship is one to one, since it is desired for rocseq_id to be a foreign key, a one-to-many relationship (crow's foot) is used to indicate that the primary key in the parent table is a foreign key in the child. (???)**
- The relationship between the PLAQUETTE table and ROC_MAP table is one to many, i.e. there can be several ROCs corresponding to a single plaquette. The same is true between the MODULE and ROC_MAP tables.
- A **rocseq_id** can be on either a plaquette or a module. This means that although a **rocseq_id** cannot take a NULL value the **plaq_id** and **mod_id** can each take on a NULL value (both cannot be NULL at the same time, however). Hence the relationship at the ROC_MAP end is optional. However, at the MODULE and PLAQUETTE ends, the relationships are mandatory since the **mod_id** and **plaq_id** cannot take on NULL values in those tables.
- For a given **rocseq_id**, the attribute **roc_pos** gives the position of the ROC within the module (**mod_id**) or the plaquette (**plaq_id**). The order of numbering is normally the same as the order in which the chips are read out.

Questions

- **In the ROC_MAP table the unique rocseq_id of a ROC is mapped to either a unique module, mod_id, or a plaquette, plaq_id. It is essential that the rocseq_id and either a mod_id or plaq_id be present. In such a scenario, are the mod_id and plaq_id attributes both marked optional?**

MODULE Table

Table Name: MODULE

Table Description:

- The **MODULE** table (entity) contains information about the modules used in the construction of the Pixel Barrel.
- The table has one entry (row) per module.

Attributes:

• **mod_id**	Unique module ID assigned at time of manufacture or an Oracle generated sequence number.
• **mod_type**	Type of module (full or half)
• **create_date**	type: date
• **create_user**	type: varchar2
• **modify_date**	type: date
• **modify_user**	type: varchar2
• **senseq_id**	Oracle generated unique sensor ID (**foreign key**)
• **hdi_id**	(**foreign key**)
• **tbm_id**	(**foreign key**)
• **mod_grade**	Grade (quality) of the module (how is this arrived at?)
• **mod_loc**	relative position of module within the ladder
• **ladder_id**	ID of ladder within a barrel layer that module is in
• **bar_half_cyl**	Half Cylinder ID that the ladder belongs to
• **bar_Zhalf**	Zhalf that the module belongs to - +Z or –Z.
• **layer**	barrel layer that module is in
• **shell**	shell that module resides in
• **sector**	sector that the module resides in

Module Table Comments

Attribute: mod_id
This is the unique module ID either assigned at time of manufacture or an Oracle generated sequence number.
No two modules can have the same mod_id, i.e. the attribute value should never be repeated.

Attribute: mod_type
There are two types of modules, full (F) and half (H). The full module has 16 ROCs while the half module has 8.

- Full barrel module ➔ **mod_type**: 0
- Half barrel module ➔ **mod_type**: 1

Attribute: senseq_id
- This is an Oracle generated unique sensor ID
- IT is a **foreign key** derived form the SENSOR table

Attribute: hdi_id
- This is a unique ID of the HDI allocated at the time of manufacture.
- The Barrel has 1 HDI per module and is mounted on top of the sensors.
- It is a **foreign key** derived from the HDI table

Attribute: tbm_id
- This is a unique ID of the TBM which is determined by the **tbmwafer_id** and the relative number within the wafer.
- The TBM is a dual chip.
- It is a **foreign key** derived from the TBM table
- A module has 1 TBM mounted on it.
- Full modules residing in layers 0 and 1 use both chips to read data out of the ROCs, 1 for 8 chips. Half modules use 1 chip.
- Full modules residing in layer 2 use only 1 chip to read out all 16 ROCs in the module.
- It may be necessary to generate a unique sequence number for the TBM as well (as in the case of ROCs and sensors).

Attribute: mod_grade

- This is the Grade (quality) of the module.
- Its value, it is assumed, will be assigned based on a series of requirements.
- These requirements have yet to be defined.
- Selection of a module for integration onto a ladder will probably be defined by the mod_grade attribute.

Attribute: mod_loc

A ladder (full/half) has 8 (full/half) modules. Modules are numbered -4, -3, -2, and -1 in the Negative Z-half. Module -4 is the farthest away from the IR. Modules are numbered 1, 2, 3, and 4 in the Positive Z-half. Module 4 is the farthest away from the IR.

Module -4	➔ mod_loc: -4
Module -3	➔ mod_loc: -3
⋮	
Module 3	➔ mod_loc: 3
Module 4	➔ mod_loc: 4

Attribute: ladder_id

Different barrels have different number of ladders – 20 in barrel 0, 32 in barrel 1, and 44 in barrel 2. The number of ladders will depend on the barrel layer, Layer. Maximum of 44

ladder_id	0	➔ ladder_id 0
ladder_id	1	➔ ladder_id 1
	⋮	
ladder_id	42	➔ ladder_id 42
ladder_id	43	➔ ladder_id 43

Attribute: bar_half_cyl

The barrel has 2 half cylinders designated right and left (with respect to the direction of the B field)

Right Half Cylinder (-X)	➔ bar_half_cyl: 0
Left Half Cylinder (+X)	➔ bar_half_cyl: 1

Attribute: bar_Zhalf

The barrel (forward) can logically be divided into 2 Z-halves

Positive Z Half (+Z)	➔ bar_Zhalf: 0
Negative Z Half (-Z)	➔ bar_Zhalf: 1

Attribute: Layer

The barrel has a maximum of 3 layers – inner, middle, and outer.

• Inner Barrel	➔ Layer: 0
• Middle Barrel	➔ Layer: 1
• Outer Barrel	➔ Layer: 2

Attribute: shell

The barrel has 4 logical shells, two in the +Z half and 2 in the –Z half.

• Shell @ (+X,+Z) quadrant	➔ Shell: 0
• Shell @ (-X,+Z) quadrant	➔ Shell: 1
• Shell @ (+X,-Z) quadrant	➔ Shell: 2
• Shell @ (-X,-Z) quadrant	➔ Shell: 3

Attribute: sector

For each Z-half, the barrel has 16 sectors. Sector 0 is on the +X side at the 12 o'clock position. The number increases in the direction of increasing ϕ so that sector 15 is on the –X side at the 12 o'clock position. The numbering scheme is the same for both Z-halves.

• sector 0	➔ sector: 0
• sector 1	➔ sector: 1
⋮	
• sector 14	➔ sector: 14
• sector 15	➔ sector: 15

MODULE_HISTORY Table

Table Name: MODULE_HISTORY

Table Description:

- The **MODULE_HISTORY** table (entity) will document any change made to the MODULE table. The documentation process will be triggered by an *UPDATE* operation performed on the MODULE table.
- The table has one entry (row) per module.

Attributes:

• **mod_hist_id**	This is an Oracle generated sequence number which will give the entry (row) a unique ID. It will also serve as the primary key of the table.
• **mod_id**	(**foreign key**) unique module ID
• **mod_hist_timestamp**	time of history record registration. It should have time with the time zone included. It together with the mod_id can constitute a unique key.
• **mod_hist_comment**	Comments (user input)
• **All attributes** (columns) of the module that is being changed	Values before the change are copied from the MODULE table and entered in the history table. In this manner, one can scan a history table and track a module through time.

Module & Module_History Comments

- A unique **mod_id** (module ID) is essential

- The unique sequence number, **mod_hist_id**, is essential since a particular module could have several entries in the history table.

- In addition to the unique **mod_id** and the components that constitute the module (**senseq_id**, **tbm_id**, **hdi_id**), the MODULE table also contains the following information
    - Location of the module within the ladder (**mod_loc**)
    - ID of the ladder (**ladder_num**)
    - Layer of the barrel the ladder is in (**barrel_layer**)
    - Sector of the barrel (**sector**)

- Having the above pieces of information packaged in the MODULE table obviates the need of additional tables, e.g. ladder and barrel. This also makes the module the final logical component in the Construction Database. The rest of the detector, ladders and barrels, derive naturally from the modules.

- A set of chips with compatible electrical properties are selected (16 or 8 + 2 or 1 extras) for module construction. This phase can be termed the start of module construction.

- A sensor is selected for mating with the selected chips. The sensor is bonded to the selected 16/8 chips (full/half module).

- Once a module has been integrated, each chip and pixel needs to be better understood. All chip measurements (described above) will need to be repeated and the new values stored in this table.

- The attributes to describe individual ROCs will all reside in a separate table, the ROC table. The attributes of the individual chip in this table will serve as a starting point for all future calibration work.

- For each ROC, a response curve for each of the DACs needs to be stored. How are these response curves made? The chip has 28 DACs. Which parameters are varied and which are kept constant when studying the response of a particular DAC?

- At this stage (after module construction), do we need to plot a S-curve to determine the threshold of each pixel?

- We also need to plot a gain curve for each pixel and may use 5 or 6 parameters to fit the curve. The calibration curve should have temperature dependence specified Calibration curve has to have temperature dependence

- Burn-in tests are performed for modules.
- Cool down to -20^0C and back to room tempereature
- Leakage current
- # of cool down cycles
- # hours tested

- Generate a bump-bond yield map.

PLAQUETTE Table

Table Name: PLAQUETTE

Table Description:

- The table will have one entry (row) per plaquette (Forward Pixel)

Attributes

• **plaq_id**	Unique plaquette ID assigned at time of manufacture or an Oracle generated sequence number.
• **plaq_type**	Type of plaquette
• **plaq_num**	Relative position of the plaquette on a panel.
• **create_date**	type: date
• **create_user**	type: varchar2
• **modify_date**	type: date
• **modify_user**	type: varchar2
• **senseq_id**	Oracle generated sequence number. It is a unique ID of the sensor on the plaquette (**foreign key**). It corresponds to a particular **senswaf_id**, **sens_id** (number within the wafer), and **stage**.
• **vhdi_id**	Unique ID of VHDI on the plaquette (**foreign key**)
• **plaq_grade**	Grade (quality) of the plaquette (how is this arrived at???)

Plaquette Table Comments

Attribute Name: plaq_id

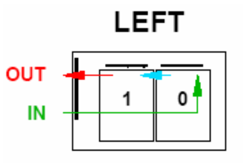
The plaq_id attribute is either a unique ID allocated to the plaquette at the time of manufacture or a unique sequence number generated by Oracle.

Attribute Name: plaq_type

- Each panel can have 3 or 4 plaquettes depending on the panel type.
- There are in all 7 types of plaquettes

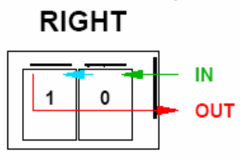
plaq_type: 0

1x2 plaquette with readout on the left



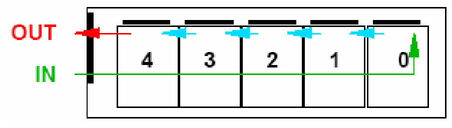
plaq_type: 1

1x2 plaquette with readout on the right



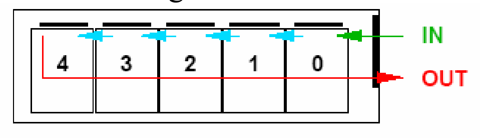
plaq_type: 2

1x5 plaquette with readout on the left



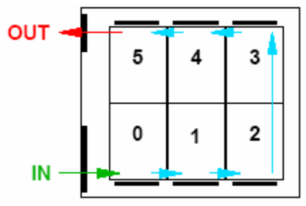
plaq_type: 3

1x5 plaquette with readout on the right



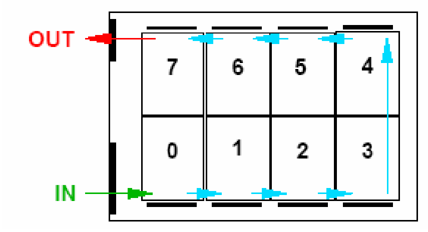
plaq_type: 4

2x3 plaquette



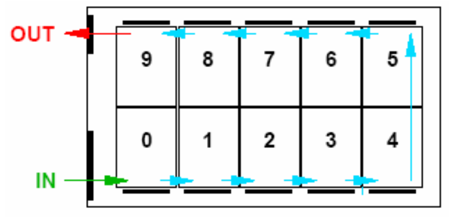
plaq_type: 5

2x4 plaquette



plaq_type: 6

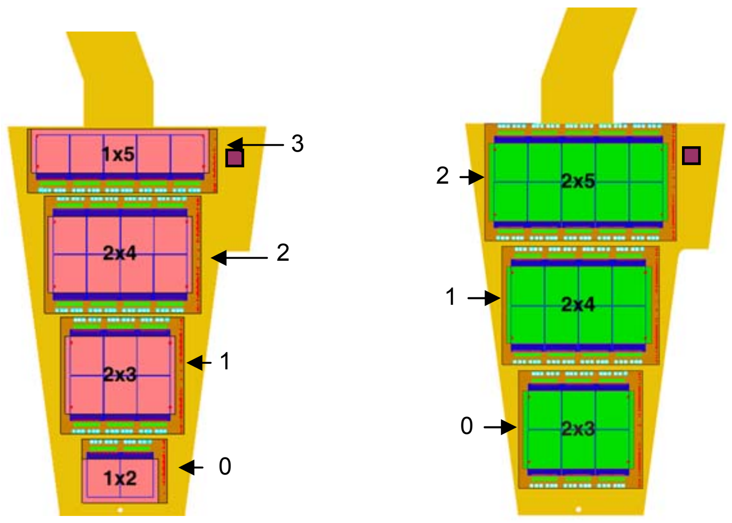
2x5 plaquette



Attribute Name: plaq_num

- On a panel, the plaquettes are numbered 0 to 3 (or 2) depending on the panel type, with 0 being closest to the beam pipe and 3 (or 2) being the farthest.

Plaquette Numbering Scheme



Attribute Name: senseq_id

- This is the unique ID of the sensor and is a foreign key derived from the SENSOR table.

Attribute Name: vhdi_id

- This is the unique ID of the VHDI and is a foreign key derived from the VHDI table.

Attribute Name: plaq_grade

- This attribute represents the grade (classification) of the plaquette.
- It is assumed that the classification is assigned based on a series of tests (yet to be specified).
- Which plaquette is selected for mounting on a panel will, it is assumed, be based on the plaq_grade attribute.

PLAQUETTE_HISTORY Table

Table Name: PLAQUETTE_HISTORY

Table Description:

- The **PLAQUETTE_HISTORY** table (entity) will document any change made to the PLAQUETTE table. The documentation process will be triggered by an *UPDATE* operation performed on the MODULE table.
- The table has one entry (row) per plaquette.

Attributes:

• **plaq_hist_id**	This is an Oracle generated sequence number which will give the entry (row) a unique ID. It will also serve as the primary key of the table.
• **plaq_id**	(**foreign key**) unique plaquette ID
• **plaq_hist_timestamp**	time of history record registration. It should have time with the time zone included. It together with the mod_id can constitute a unique key.
• **plaq_hist_comment**	Comments (user input)
• **All attributes** (columns) of the plaquette that is being modified	Values before the change are copied from the MODULE table and entered in the history table. In this manner, one can scan a history table and track a module through time.

PANEL Table

Table Name: PANEL

Table Description:

The table will have one entry (row) per panel.

Attributes

• **pan_id**	Unique panel ID assigned at time of manufacture or an Oracle generated sequence number
• **pan_type**	Type of panel (total of 8 types, i.e. 4 pair types)
• **create_date**	type: date
• **create_user**	type: varchar2
• **modify_date**	type: date
• **modify_user**	type: varchar2
• **blade**	ID of blade that panel is mounted on (0 → 23)
• **disk_face**	6 disk_faces: 0, 1, 2, 3, 4, 5 on either side of IR
• **disk**	physical disk position (values: 0,1,2 – 0 closest to IR and 2 farthest from IR)
• **disk_group**	+Z (0) along +B or –Z (1) section of detector
• **fwd_half_cyl**	forward half cylinder, right half (0) or left half (1)
• **hdi_id**	(**foreign key**)
• **tbm_id**	(**foreign key**)
• **pan_grade**	Grade of the panel

PANEL Table Comments

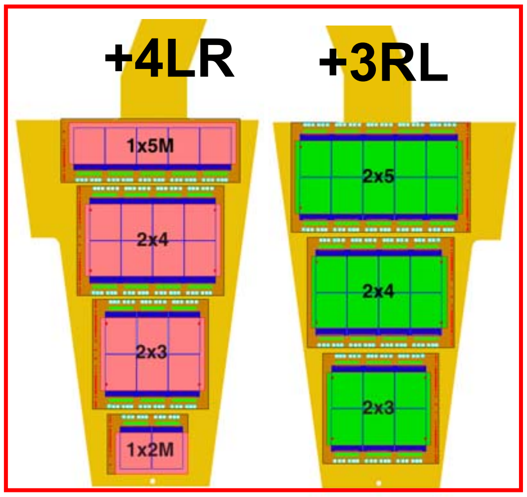
Attribute Name: pan_type

Depending on the location of the blade (on the disk), the type of the panel(s) on a blade can be different. There are in total 8 different types of panels.

- Panels with 4 plaquettes (type: 0, 2, 4, 6) always face toward the IR while the ones with 3 plaquettes (type: 1, 3, 5, 7) reside on the other side of the blade.
- Panels (type: 0, 1, 2, 3) reside on the +B (or +Z) side of the IR
- Panels (type: 4, 5, 6, 7) reside on the –B (or –Z) side of the IR

The 8 different panel types are group in 4 different pairs – 01, 23, 45, & 67

pan_type 0	➔ 4RR, near, +B, Q1 and Q3
pan_type 1	➔ 3LL, far, +B, Q1 and Q3
pan_type 2	➔ 4LR, near, +B, Q2 and Q4
pan_type 3	➔ 3RL, far, +B, Q2 and Q4
pan_type 4	➔ 4RL, near, -B, Q1 and Q3
pan_type 5	➔ 3LR, far, -B, Q1 and Q3
pan_type 6	➔ 4LL, near, -B, Q2 and Q4
pan_type 7	➔ 3RR, far, -B, Q2 and Q4



Q ➔ quadrants

Q1: (+X, +Y)
Q2: (+X, -Y)
Q3: (-X, -Y)
Q4: (-X, +Y)

Attribute Name: blade

- Each disk is made up of 24 turbine blades. A blade has 2 panels, one on each face.
- In the scheme used, the blades are numbered in an increasing order in the direction of increasing phi, i.e. anticlockwise, with the phi-direction along the +Y axis of the coordinate system employed.
- Blades 0 through 11 reside in the Left Half Cylinder and blades 12 through 23 reside in the right half cylinder (for both +Z and –Z).
- Each blade is rotated 20^0 clockwise about its axis in the +Z half and anticlockwise in the –Z half.
- A panel is numbered the same as a blade, but is also dependent on which disk_face it is resident on.
- Another way to view a panel is as a member of a disk_face. A disk_face has 24 panels.

    - blade = 0, 1, 2, . . . , 23

Attribute Name: disk_face (logical disk)

Each disk has 2 faces. Hence there is a total of 6 faces per disk_group.
Within each group,
disk 0 contains disk_faces 0 and 1
disk 1 contains disk_faces 2 and 3
disk 2 contains disk_faces 4 and 5

disk_faces 0, 2, and 4 face towards the IR and disk_faces 1, 3, and 5 face away from the IR.
Within each disk_group, disk_face = 0, 1. . . 5

Attribute Name: disk

There are 3 disks in a disk_group (one on each side of the IR)
Each Forward Disk Group has 3 disks (6 faces). Detectors (sensors & readout chips) are mounted on faces.
Within each disk group,

- disk 0 is the closest to the IR
- disk 2 is the farthest from IR

Attribute Name: disk_group

There are 2 Forward Pixel Disk groups, one in the +B direction and the other in the –B direction of the IR.
disk_group 0 is on the +B (+Z) side of the IR
disk_group 1 is on the –B (-Z) side of the IR

Attribute Name: fwd_half_cyl
There are 2 half cylinders designated right and left (with respect to the direction of the B field) in the forward pixel detector.

- Right Half Cylinder ➔ fwd_half_cyl: 0
- Left Half Cylinder ➔ fwd_half_cyl: 1

Attribute Name: hdi_id (Forward High Density Interconnect ID)

- Each HDI has a unique ID (allocated at time of manufacture) represented by the attribute hdi_id.
- It is a foreign key from the HDI table.
- The HDI is glued to a Beryllium panel.

hdi_id: unique ID of a HDI

Attribute Name: tbm_id

- This attribute represents a unique ID of the TBM on the panel.
- The unique ID could be a combination of the ID of the wafer it is derived from and the relative position of the chip on the wafer.
- It is a foreign key from the TBM table.
- A panel has 1 TBM

Attribute Name: pan_grade

- This attribute represents the grade (classification) of the panel.
- The requirements to set a value for this attribute are not defined yet.
- It is presumed that this attribute will contribute to the selection of a panel for mounting on a disk.

PANEL_HISTORY Table

Table Name: **PANEL_HISTORY**

Table Description:

- The **PANEL_HISTORY** table (entity) will document any change made to the **PANEL** table. The documentation process will be triggered by an *UPDATE* operation performed on the MODULE table.
- The table has one entry (row) per panel.

Attributes:

• **pan_hist_id**	This is an Oracle generated sequence number which will give the entry (row) a unique ID. It will also serve as the primary key of the table.
• **pan_id**	(**foreign key**) Unique panel ID
• **pan_hist_timestamp**	time of history record registration. It should have time with the time zone included. It together with the **pan_id** can constitute a unique key.
• **pan_hist_comment**	Comments (user input)
• **All attributes** (columns) of the panel entity (row) that is being changed.	Values before the change are copied from the PANEL table and entered in the history table. In this manner, one can scan a history table and track a panel through time.

PLAQ_PAN_MAP Table

Table Name: PLAQ_PAN_MAP

Table Description:

Multiple **plaquettes** are mounted on a **panel**. The **PLAQ_MAP** table is a mapping table that will enable a rapid correlation of a panel with the plaquettes mounted on it.

There is one entry (row) for each individual plaquette identified by **plaq_id**. Each row (entry) will contain two unique attributes of which one is the **plaq_id** and the other **pan_id**, the ID of the panel on which the plaquettes are mounted.

Attributes:

• **plaq_id**	(**foreign key**) The unique plaquette ID.
• **pan_id**	(**foreign key**) The unique panel ID
• **plaq_pos**	position of the plaquette within a panel
• **create_date**	
• **create_user**	
• **modify_date**	
• **modify_user**	

Comments

- It is the plaquettes that are subjected to burn-in tests. What parameters are measured during this phase and what should be stored in the database?

- **plaq_id**: A unique plaquette ID is essential. This attribute constitutes the primary key. It should not take on a NULL value and any value it takes on should not be repeated within the table. It is assumed that the plaquettes will be allocated a unique ID number during the manufacturing process.

- **plaq_type**: There are in all 7 types of plaquettes

plaq_type = 0:	PLAQ_1x2L	(readout in left)
plaq_type = 1:	PLAQ_1x2R	(readout in right)
plaq_type = 2:	PLAQ_1x5L	
plaq_type = 3:	PLAQ_1x5R	
plaq_type = 4:	PLAQ_2x3	
plaq_type = 5:	PLAQ_2x4	
plaq_type = 6:	PLAQ_2x5	

- Panel types (pairs):

+4LR	+Z side, 4 plaquettes, porch on left side, pig tail bends right
+3RL	+Z side, 3 plaquettes, porch on right side, pig tail bends left
-3RR	
-4LL	
+4RR	
+3LL	
-4RL	
-3LR	

- A panel consists of multiple plaquettes. Correlation between a panel and the plaquettes mounted on it is carried out using the PLAQ_MAP table.

- A unique **pan_id** (module ID) is essential. The unique sequence numbers, **plaq_hist_id** and **pan_hist_id**, are also essential since a particular module or plaquette could have several entries in the associated history tables.

- In addition to the unique **pan_id** and the components that constitute the module (**plaq_id**s, **tbm_id**, and **hdi_id**), the PANEL table also contains the following information
    - Blade number that panel resides on (**blade**)
    - ID of the disk (**disk**)
    - Face of the disk (**face**)
    - Section of the forward (**section: +Z or -Z**)

  The **plaq_id**s and **plaq_pos** within a panel are from the PLAQ_MAP table.

- Having the above pieces of information packaged in the PANEL table (just as the MODULE table) obviates the need of additional tables, e.g. blade and disk. This also makes the panel the final logical component in the Construction Database. The rest of the detector, blades and disks, derive naturally from the panels.

- A set of chips with compatible electrical properties are selected for plaquette and panel construction. In the case of the Forward Pixel, this has to be performed in two sections, the ROC_MAP table where chips are selected for plaquette construction and the PLAQ_MAP table where the plaquettes are selected for mounting on a particular panel.

- Once a plaquette has been integrated, each chip and pixel needs to be better understood. All chip measurements (described above) will need to be repeated and the new values stored in this table.

- The attributes to describe individual ROCs will all reside in a separate table, the ROC table. The attributes of the individual chip in this table will serve as a starting point for all future calibration work.

- For each ROC, a response curve for each of the DACs needs to be stored. How are these response curves made? The chip has 28 DACs. Which parameters are varied and which are kept constant when studying the response of a particular DAC?

- At this stage (after module construction), do we need to plot a S-curve to determine the threshold of each pixel?

- We also need to plot a gain curve for each pixel and may use 5 or 6 parameters to fit the curve. The calibration curve should have temperature dependence specified.

    - Burn-in tests are performed for plaquettes.
    - Should burn-in tests be also performed for panels?
    - Temperature cycle: Cool down to -20^{0}C and back to room temperature
        - Leakage current
        - # of cool down cycles
        - # hours tested
    - Generate a bump-bond yield map.

Communication Network Hub Table

Table Description

Attributes

Create Date	(updated when an entry is made in the database)
Create User	(updated when an entry is made in the database)
Modify Date	(updated when an already entered attribute is altered)
Modify User	(updated when an already entered attribute is altered)
Oracle Sequence Number	(Updated by Oracle after entries made in the database)
Wafer ID	
CNHub ID	

Questions & Issues

What parameters are used to test the device?
What parameters are used for quality selection?

Equipment Management Database

It is quite plausible that the Construction Database will evolve into the Equipment Management Database. The Equipment Management Database will incorporate a substantial amount of information from the Construction Database and some additional information.

The goal of the Equipment Management Database is to keep track of the smallest replaceable unit in the detector.

- What is the smallest replaceable unit in the detector? – Module???
- Needed to keep track of the smallest replaceable unit. Required by INB specifications to keep track of radioactive material.
- Electronics boards: all hardware in USC (underground service cavern) or UXC (underground experimental cavern).
- If a replaceable unit is detached from the detector (it becomes a child) and because of the strict radioactive regulations, it must be tracked very closely. This database should keep track of the child.
- Keep track of fibers, cable trays, etc.
- The database does not have to keep track of small components that can be simply dropped into a bucket, e.g. nuts and bolts.

Configuration Database

The configuration database stores entities to start up a run and has the potential of becoming a very large database. It should be accessible to DAQ, DCS (controls), and the detector (for monitoring) systems. This database should contain per-pixel as well as per-chip information that is necessary to bring each individual pixel and readout chip to operational state. It should also contain information for other components necessary for data taking.

Per Pixel Database entry (Configuration Database)

- Large number of pixels - ~ 66M or 73M
- Each entry will have
  - Detector ID (barrel, B+ disks, or B- disks) stored as **1 byte**
  - Module # stored as **4 bytes**
  - ROC ID stored as **4 bytes**
  - Pixel ID
    - double column # stored as **1 byte**
    - row # stored as **1byte**
  - Thresholds (trim bits)

    3 bit thresholds, downloaded to the front-end. Stored in FEC modules and archived in the DCS Configuration database. There will be 1 to 2 new sets per day.

    Data: 3 bits stored as 1 byte
  - Mask bit

    1 bit mask is downloaded to the front-end. Stored in FEC modules and archived in the Configuration database.

    Data: 1 bit stored as 1 byte
  - Pedestals

    Measured and stored 1-2 per day. They are stored in the pixel monitoring system and in the DCS configuration database.

    Data: Stored as 4 bytes
  - Gain (charge/count)

    The gain is determined 1 to 2 times per day. It is stored in the pixel monitoring system and in the Configuration database. It is used to convert ADC counts to electrons.

    Data: Stored as 4 bytes
  - Efficiencies

It is a calculated parameter used to monitor detector performance. It is stored 1 to 2 times a day in the monitoring system and in the Configuration database.
Data: Stored as 4 bytes

- Average rates
  It is a calculated parameter used to monitor detector performance. It is stored 1 to 2 times a day in the monitoring system and in the Configuration database.
  Data: Stored as 4 bytes

Data per pixel (as we know): 29 bytes
For all estimates, use 32 bytes
The number could grow

Storage for 1 calibration run: ~ 2 GB (66M pixels)
~ 2.4 GB (73M pixels)

@ 2 times a day, max storage required: ~ 4 GB (66M pixels)
~ 5 GB (73M pixels)

Per Readout Chip Database Entry (Configuration Database)

- Total number of ROC's ~ 15840 (3 barrels, 4 disks configuration)
  ~ 17500 (3 barrels, 6 disks configuration)

- Each entry will have

  - 28 DAC settings - 28 bytes (1 byte per DAC)
  - 2 Control registers - 8 bytes (4 bytes per register)
  - Temperature - 4 bytes

Total per data collection: ~ 634 KB (4 disks)
~ 700 KB (6 disks)

Other Configuration Database entries could include those related to

- Sensors
- Electronics
- Cables/fibers & corresponding calibration parameters
- Cooling/flow parameters

- The Configuration Database has the potential of being extremely large. At ~ 10 GB of data a day, the storage requirement alone will amount to ~ 3 TB or more a year. Database storage becomes a problem in itself. We will need to decide what portion of the database will continue to remain online and what will be archived.

- The sheer volume of data needed to be accessed in order to start up a run could pose a serious problem in incurred access overhead. There is first the problem of data access from the database (storage) to the server memory. This in itself is a fairly slow process that will be driven by the speed of the storage media. Then there is the transfer of the retrieved data to the requesting clients, possibly processes running in a Linux workstation. Once in the memory of the Linux workstation(s), data has to be downloaded to the memory of the FEC's via the VME backplane(s) after which the FEC's will proceed to transfer data to the appropriate ROC's via the designated TBM's using PSI^{2}C protocol. Another complication is that data retrieved from the database have to be sent to different destination, e.g. FEC, monitoring system, etc.

- There is also the scenario of a monitoring process having to automatically deal with hot channels in the course of a run with minimal disruption to the data-taking process. This will require the monitoring process to update the mask bit, threshold bits, etc. of the detected hot channel in the FEC memory as well as the master database. We will have to make sure the corresponding mask and trim bits are altered in the database and the related FEC memory before issuing an I^{2}C command to change the bit of the offending pixel. This then raises several questions
    - How is pixel monitoring going to be done?
    - What parameters need to be actively tracked?
    - How are hot pixels going to be masked?
    - What kind of human interface to the monitoring system are we likely to have?

- Overall the Configuration Database has the potential of becoming very complicated. We need to think the process over very well prior to a final decision.

- Besides the entries discussed above, several other entries may be necessary in the Configuration Database

    - **Sensor Database Entry**
    - **Module Database**
    - **Electronics Database**
    - **Cooling Monitor Database**
    - **Hot Pixel Monitor Database**
    - **Etc.**

Conditions Database

What the Conditions Database (also the Calibration Database) will contain is not quite clear. It will definitely contain **calibration** and **alignment** data that are necessary for both online and offline reconstruction.

Given the large number of pixels, the Conditions Database also has the potential of becoming extremely large. A calibration set (as defined by Danek) as we currently know consists of

- Thresholds @ 3 bits/pixel
  May require a full download 3-4 per day

- Pedestals @ 4 bytes/pixel
  Measured stored 1-2 per day
- Gains @ 4 bytes/pixel (how?)
  Determined 1-2 per day
- Performance parameters
  Efficiencies @ 4 bytes
  Average rates @ 4 bytes
    Stored 1-2 per day

We do have to remember that this is most likely the information that will be needed for event reconstruction in Level2 as well as Level3 (when it is decided). We are again talking about a substantial amount of data transfer from the Conditions Database.